DOI:
10.1016/j.engappai.2015.03.008

Document Version
Peer reviewed version

Decision Making with Natural Language based Preferences and Psychology-inspired Heuristics

Ingrid Nunes[a,*], Simon Miles[b], Michael Luck[b], Simone Barbosa[c], Carlos Lucena[c]

[a] *Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil*
[b] *King's College London, Strand, London, WC2R 2LS, United Kingdom*
[c] *Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil*

Abstract

Decision making is required by many tasks, such as shopping, nowadays assisted by software systems, and providing support to the decision making process is a feature that would significantly improve such systems. Many decision support systems and related approaches have been proposed to that purpose, but they often involve tedious elicitation processes or previously collected data. In this paper, we propose an automated decision making technique, which chooses an option from the set of those available based on preferences and priorities expressed in a high-level preference language, exploiting natural-language terms, such as expressive speech acts. Moreover, in order to make a decision, our technique goes beyond the provided preferences with psychology-inspired heuristics, which concern how humans make decisions, as provided preferences are typically not enough to resolve trade-offs among available options. Two studies were performed to evaluate our approach, and results indicate that our technique is effective both by comparing its recommendations with those made by a human expert, and by considering evaluation scores provided by users that experienced our technique.

Keywords: Decision Making, Preference Reasoning, Natural Language based Preferences, Psychology Heuristics

*Corresponding author.
Email addresses: ingridnunes@inf.ufrgs.br (Ingrid Nunes), simon.miles@kcl.ac.uk (Simon Miles), michael.luck@kcl.ac.uk (Michael Luck), simone@inf.puc-rio.br (Simone Barbosa), lucena@inf.puc-rio.br (Carlos Lucena)

1. Introduction

Making decisions over a possibly huge set of options is part of many of the tasks that people face in their everyday life, such as choosing products to buy, where to go for fun or what to eat. Choosing from a set of available options often requires resolution of trade-offs, but it can be unfeasible for humans to carefully evaluate each option of a large set due to the required amount of time and cognitive effort, so that they are often unsatisfied with their choices (Schwartz, 2005). Since understanding human decision making and how to support them in making choices is important from many perspectives, such as understanding consumer behaviour and aiding managers in making high-impact business decisions, decision making has been extensively studied in a wide range of areas, including *economics* (Keeney and Raiffa, 1976), *marketing* (Wierenga, 2008), and *psychology* (Tversky, 1996), for many decades. Moreover, receiving support or delegating a decision to a software system that is aware of users' preferences and is able to make decisions like them can be very helpful (Kaklauskas et al., 2007), because computers are able to process a large set of options in a relatively short time. Therefore, decision making has also received much attention in many areas of *computer science*, including artificial intelligence, databases, and semantic web.

The many proposed approaches to support decision making can be split into two groups. The first group takes as input ratings given to options or pairwise comparisons and, using machine learning algorithms, predicts preferred products (Adomavicius and Tuzhilin, 2005; Domshlak and Joachims, 2007). The main advantage of such approaches is that users may receive recommendations without explicitly providing information. However, approaches in this group require, sometimes a significant amount of, previously collected data. Moreover, the reason why a product is considered preferred is hidden in statistical calculations of machine learning techniques, thus preventing the provision of explanations for a recommendation, which are essential for users to accept it (Shafir et al., 1998). The second group, which includes multi-criteria decision analysis (MCDA) (Greco, 2005; Wan and Li, 2013; Wan and Dong, 2014) and qualitative preference reasoning (Domshlak, 2008) approaches, builds a user preference model and derives the best options. MCDA requires an elicitation process, which makes users to perform pairwise comparisons in order to identify a utility function (UF) (Keeney and Raiffa, 1976) that represents the user preferences. Qualitative preference reasoning, in turn, takes as input qualitative preference statements often over option attributes and, through algorithms that reason about such preferences or transform them into a UF, selects options that are considered opti-

mal, e.g., (Boutilier et al., 2004). Although such approaches allow extraction of a rationale behind the choice, they require having enough information to choose among options, i.e. the user preference model must capture the trade-offs between options, which is information that users do not provide unless they are asked. According to Lichtenstein and Slovic (2006), humans have a set of preferences that they are aware of — referred to as *known* preferences — which guide the decision making process. However, additional preferences to resolve trade-offs are constructed (as opposed to revealed) during the decision making process, as "*people do not maximise a precomputed preference order but construct their choices in the light of available options*" (Tversky, 1996). Because trade-off resolution requires cognitive effort of users (Schwartz, 2005), this compromises the acceptance of decision support systems that use elicitation processes based on questions that ask users to make pairwise comparisons that require them to resolve trade-offs.

We propose a novel technique to support decision making. It consists of a set of algorithms that take as input user preferences over option attributes — like the second group of approaches — but include heuristics inspired from psychology (Shafir et al., 1998; Simonson and Tversky, 1992) to derive a quantitative preference model, in the form of a decision function similar to a UF. Such decision function allows choice of an option from a set available according to provided user preferences. The reason for taking into account such heuristics is to make a decision similar to that made by users without requiring them to provide preferences that resolve trade-offs. Moreover, our goal is to allow users to freely express their known preferences — existing work on preference reasoning is only able to handle a restricted set of preference types, thus constraining users in expressing their preferences. In our technique, provided user preferences are expressed in a high-level preference language, which include natural-language-like expressions, such as expressive speech acts (e.g. *like*, *accept* or *need*). Such preferences are compiled into two computational models that facilitate preference processing. These models are the Preference Satisfaction Model (PSM), which combines the information given by monadic preferences (those that have a single referent), and the Options-Attribute Preference Model (OAPM), which indicates which attribute value is better considering each two options. Based on these computational models, we cannot directly conclude which available option is the "best" or decide which of two options is better, but they expose positive and negative aspects of available options, integrating the information provided by heterogeneous types of preferences. As heuristics consist of an approximation, our ultimate goal is to use the proposed approach in a scenario in which users: (i) provide a set of high-level preferences; (ii) receive a recommendation with associated explanations; and (iii)

refine their preferences according to the provided explanations (Klein and Shortliffe, 1994; Labreuche, 2011; Nunes et al., 2014). In particular in this paper, we address two problems associated with this scenario: how to handle high-level preferences, and how to make a decision using them and psychology-inspired heuristics. Two studies were performed to evaluate our approach, and results indicate that our technique is effective both by comparing its recommendations with those made by a human expert, and by considering evaluation scores provided by users that experienced our technique.

The remainder of this paper is organised as follows. Section 2 overviews our decision making technique. Section 3 presents the high-level preference language that our technique is able to process, and a running example that will be used throughout the paper. Next, the steps of our technique are detailed in Section 4, and its evaluation is described in Section 5. Finally, Section 6 presents related work, and Section 7 concludes.

2. Technique Overview

Our decision making technique is composed of a set of algorithms that together solve a decision problem. A decision problem consists of choosing one option o based on preferences from a finite set of available options, Opt, where all $o' \in Opt$ are of the same class (or type), for example a set of *hotels*. Each class is associated with a finite set of attributes, Att, and each $a_i \in Att$ is associated with a domain D_i, which establishes the values allowed for that attribute. Each domain D_i: (i) consists of a set of values x_{ij}; (ii) can be *discrete* or *continuous*; and (iii) can be *ordered* or *unordered*. For example, real numbers are an ordered continuous domain, integers are an ordered discrete domain, and colours are an unordered discrete domain. We refer to domains composed of numbers as *numeric*.

In our work, we have a set of *assumptions*. First, users have a set of known preferences over the problem for which the decision has to be made, and they are able to express them in a high-level language. Second, we consider a *single decision maker*; that is, provided preferences are given by a single user, and our goal is to make a choice that maximises the satisfaction of this user. Third, we assume a *consistent* set of preferences. Users may provide conflicting preferences, which are those that cannot be achieved at the same time, e.g. "I prefer higher quality and lower price." However, we assume they do not provide inconsistent preferences, for example, "I prefer A to B, and I prefer B to A." Finally, available options, attributes and their domains are *given* and are, therefore, inputs of our technique, together with user preferences.

The goal of our decision maker is to simulate human reasoning in making decisions, allowing us to exploit natural user expressiveness of preferences (without the need for elicitation methods) and to resolve trade-offs (that cannot be resolved only with the provided preferences) in a way humans would do. The overall decision making process is inspired by *reason-based choice* (Shafir et al., 1998), which states that people make decisions by identifying reasons to accept and reject options. In addition, it incorporates two heuristics (Simonson and Tversky, 1992): *extremeness aversion* (avoiding extreme options, which are those that compensate large gains with large losses), and *trade-off contrast* (influencing the preference between two options with the cost-benefit relationship of all options).

We introduce our technique in Figure 1. This figure shows the four steps of the technique (indicated with dashed lines), and their activities together with inputs and outputs — the input of one activity may be the output of another. First, it can be seen that one group of preferences — monadic preferences, which indicate preferences with expressive speech acts or ratings — are used by many activities of our technique, showing that the technique is *driven by these natural-language expressions*. Second, the technique has *variable parts*: our technique uses a particular interpretation of natural language expressions, but this interpretation can change and be customised to specific applications. Moreover, as we discuss later, during the decision making process our technique calculates quantitative costs of options based on qualitative preferences, and different functions associated with this calculation can be adopted. Based on experimentation, we selected particular instances (adopted in this paper) for these parts. Third, our technique is composed of *four main steps*, explained next.

Pre-processing. Our preference language allows the expression of heterogeneous types of preferences. The pre-processing step involves building computational models that compile information given by the different preferences provided by users and represent options in such a way that their positive and negative aspects are made explicit (according to those preferences).

Explication. Sometimes a preference provided by a user implies a further preference in addition to its literal meaning. So, in the explication step we consider *implicit* preferences that we can extract from the preferences explicitly given by users, and based on this information we update the previously produced computational models.

Elimination. When people make a choice from a set, they first eliminate options that have no advantage when compared to another, i.e. *dominated* op-

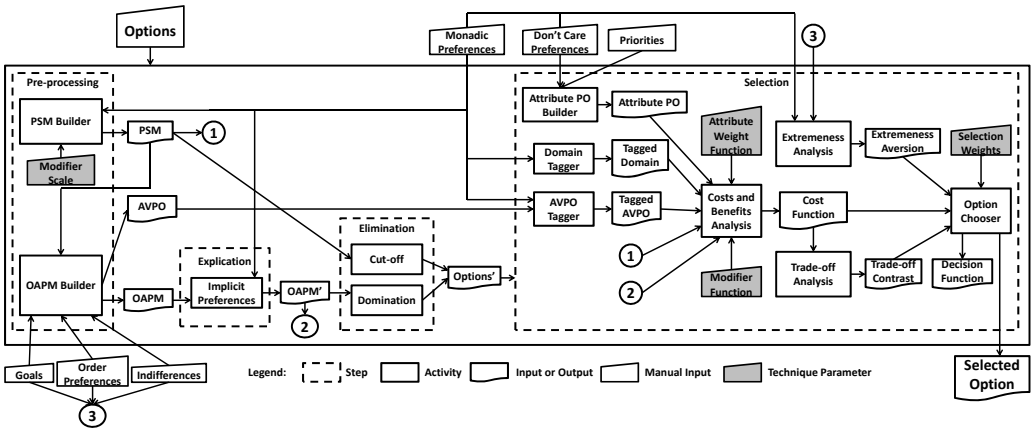

Figure 1: Technique Overview.

tions, or have attributes with unacceptable values, i.e. non-compensatory attributes. In the elimination step, we discard these two kinds of options.

Selection. After eliminating the options above, we obtain a *consideration set*, which contains options that require trade-off resolution to make a choice. In order to make this selection, we first analyse option costs and benefits, by using the information compiled in our computational models to calculate each option's costs with respect to another for each attribute. The overall cost of an option (w.r.t. another) is then a weighted sum of these individual attribute costs, derived from provided priorities. Next, the trade-off between options and how these options compensate advantages with disadvantages are analysed (which are related to the trade-off contrast and extremeness aversion principles), and these factors are them combined with the previously calculated option costs.

The output of our technique is a partially ordered set, organised in four levels: (i) *chosen option*, which is the (purported) optimal option; (ii) *acceptable options*, which are in the consideration set, but were not chosen; (iii) *eliminated options*, which were discarded because of non-compensatory attribute(s); and (iv) *dominated options*.

Although our technique has variable parts instantiated with perhaps not the best possible instances, we already achieve good results according to our evaluation, but we aim to further improve results by exploring this variability. Note that making a decision based on *heterogeneous* preferences, like those presented

in the next section, in a *consistent* way is a challenge, and this is an important contribution of our work, together with a way of computing psychology-inspired heuristics. Moreover, our technique decomposes the decision making problem based on these heterogeneous preferences in many steps and activities in a modular way, so that improvements can be made to different parts separately, such as how natural-language expressions are interpreted and the choice for the different functions of our variable parts. Before describing each step of our technique, we next introduce the preference language in which provided user preferences must be expressed to be used as input of our technique.

3. Preference Language and Running Example

There are different forms in which humans express preferences, and our goal is to provide them with a language in which they can express their preferences in a way as close as possible to natural language. Preferences can be *monadic* or *dyadic* (Hansson, 2001), where the former evaluates a single referent, e.g. "*I like skiing*," and the latter indicates a relation between two referents, e.g. "*I prefer skiing to surfing*." Based on a previous study involving almost 200 preference specifications (Nunes et al., 2010, 2013), we derived a high-level preference language, whose EBNF is presented in Table 1, which includes seven types of preferences and means for specifying priorities among attributes and preferences. While constraints (e.g. *price lower than £105*), qualifying (e.g. *I hate brand A*) and rating (e.g. *brand A is the best*) are monadic preferences; goals (e.g. *minimise price*), orders (e.g. *brand A is preferred to brand B*) and indifferences (e.g. *brand A and brand B are equally preferred*) are dyadic. Preferences expressed using the expressions *require*, *need*, *hate* and *don't accept* are considered hard constraints (but this is subject to particular interpretations).

Preferences are restricted to refer to only one attribute (this restriction is not extended to conditions). As a consequence, propositional formulae of constraints cannot refer to different attributes, e.g. "*I prefer a laptop with a 15" screen and integrated camera*" (which can be decomposed into two preferences). Moreover, order statements, as shown by the language grammar, refer only to expressions and not constraints. Some of the preferences not allowed by our restrictions can be expressed in a different allowed manner, e.g. "*I prefer a laptop with a 14" or 15" screen to one with a 17" screen*" can be expressed with two preferences: "*I prefer a laptop with a 14" screen to one with a 17" screen*" and "*I prefer a laptop with a 15" screen to one with a 17" screen*."

$$
\begin{aligned}
\mathit{preference} &::= [\mathit{condition}]\ (\mathit{constraint} \mid \mathit{qualifying} \mid \mathit{rating} \mid \mathit{goal} \mid \mathit{order} \\
&\quad \mid \mathit{indifference} \mid \mathit{dontCare}) \\
\mathit{condition} &::= \textbf{if}\ \mathit{formula}\ \textbf{than} \\
\mathit{formula} &:: \mathit{expression} \mid \mathit{formula}\ \textbf{and}\ \mathit{formula} \mid \mathit{formula}\ \textbf{or}\ \mathit{formula} \\
&\quad \mid \textbf{not}\ \mathit{formula} \\
\mathit{expression} &::= \mathit{attribute}\ (= \mid \neq \mid > \mid \geq \mid < \mid \leq)\ \mathit{value} \\
\mathit{constraint} &::= \mathit{formula} \\
\mathit{qualifying} &::= \mathit{expressive\_speech\_act}\ \mathit{formula} \\
\mathit{rating} &::= \mathit{formula}\ \mathit{rate} \\
\mathit{goal} &::= (\textbf{minimise} \mid \textbf{maximise})\ \mathit{attribute} \\
\mathit{order} &::= \mathit{attribute} = \mathit{value} \succ \mathit{attribute} = \mathit{value} \\
\mathit{indifference} &::= \textbf{indifferent}\ \mathit{formula}\ \{\mathit{formula}\} \\
\mathit{dontCare} &::= \textbf{dont\_care}\ \mathit{attribute} \\
\mathit{expressive\_speech\_act} &::= [\textbf{don't}]\ (\textbf{prefer} \mid \textbf{need} \mid \textbf{desire} \mid \textbf{avoid} \mid \textbf{like} \mid \textbf{want} \\
&\quad \mid \textbf{accept} \mid \textbf{require} \mid \textbf{love} \mid \textbf{hate}) \\
\mathit{rate} &::= \textbf{best} \mid \textbf{very\_good} \mid \textbf{good} \mid \textbf{neutral} \mid \textbf{bad} \mid \textbf{very\_bad} \mid \textbf{worst} \\
\mathit{priority} &::= [\mathit{condition}]\ (\mathit{attribute\_priority} \mid \mathit{attribute\_indifference} \\
&\quad \mid \mathit{preference\_priority}) \\
\mathit{attribute\_priority} &::= \mathit{attribute} \rhd \mathit{attribute} \\
\mathit{attribute\_indifference} &::= \mathit{attribute} \sim \mathit{attribute} \\
\mathit{preference\_priority} &::= \mathbb{Z}.\ \mathit{preference}
\end{aligned}
$$

Table 1: Preference Language Syntax.

In order to illustrate our high-level preference language, we introduce an example in this section. Throughout this paper, this example will also be used to illustrate different parts of our decision making technique. Suppose *Bob* is visiting a university, and needs to choose an apartment to stay at. Each apartment is described in terms of *seven* attributes, described in Table 2, each associated with a domain. Bob's preferences are shown in Table 3, using our preference language. Preferences are numbered (1...15) because of preference priorities. The final line is an attribute priority. These preferences and priorities are used to make a choice on Bob's behalf. The decision problem is to choose one apartment from the available options, shown in Table 4. We describe next the steps of our technique.

Table 2: Apartment Attributes.

Attribute	Domain	Description
uni	$\{x \mid x \in \mathbb{R}, 0 < x \leq 15\}$	The distance, in kilometres, from the apartment to the university.
station	$\{x \mid x \in \mathbb{R}, 0 < x \leq 1.2\}$	The distance, in kilometres, from the apartment to the closest underground station.
market	$\{x \mid x \in \mathbb{R}, 0 < x \leq 0.7\}$	The distance, in kilometres, from the apartment to the closest supermarket.
zone	$\{x \mid x \in \mathbb{R}, 0 < x \leq 0.7\}$	The underground coverage in the city in which the university is located is split into six zones. Zone 1 is the city centre, and the higher the zone number is, the farther it is from the centre.
brand	$\{A, B, C, D\}$	Each apartment has a brand, associated with the company to which it belongs.
start	$\{x \mid x \in \mathbb{Z}, 1 \leq x \leq 5\}$	A number that represents the apartment quality; the higher, the better.
price	$\{x \mid x \in \mathbb{R}, 95 \leq x \leq 125\}$	The price of renting the apartment (per week).

4. Technique Steps

4.1. Pre-processing

In our approach, the first step to make a decision is to pre-process the preferences provided by users and analyse them according to the available options. As previously introduced, there are monadic and dyadic preferences, and we process them separately, building two models based on them — the Preference Satisfaction Model (PSM) (Section 4.1.1) and the Options-Attribute Preference Model (OAPM) (Section 4.1.2).

4.1.1. Preference Satisfaction Model

The first model, named *Preference Satisfaction Model (PSM)*, consists of a table that captures how options satisfy preferences in terms of each attribute according to monadic preferences. This table associates option attributes with an expressive speech act (or their negation) or a rate, meaning that the preference for an attribute value of a particular option is represented by a specific expressive speech act or rate. For example, the PSM value of option Ap_A and attribute *price* is $\langle \epsilon, require \rangle$, because the *price* of option Ap_A satisfies *require* (Preference 5). $s \in ExpressiveSpeechActs$ is an expressive speech act that comes from qualifying preferences, while $r \in Rates$ is a rate (e.g. *love* and *hate*) that comes from rating preferences. Expressive speech acts and rates are collectively referred to as *modifiers* ($M = ExpressiveSpeechActs \cup Rates$). Constraint preferences, which are associated with no modifier, are considered to be associated with an implicit modifier, namely *want*. All these preferences are referred to as monadic preferences ($MP = Constraints \cup QualifyingPreferences \cup RatingPreferences$). The PSM is defined as follows.

1. **don't accept** *zone* > 2
2. **prefer** *uni* ≤ 2.5*Km*
3. **if** *uni* ≤ 2.5*Km* **then need** *station* ≤ 1*Km*
4. **if** *uni* ≤ 2.5*Km* **then prefer** *station* ≤ 0.7*Km*
5. **if** *uni* ≤ 2.5*Km* **then require** *price* ≤ £125
6. **if** *uni* > 2.5*Km* **then need** *station* ≤ 0.7*Km*
7. **if** *uni* > 2.5*Km* **then require** *price* ≤ £105
8. **minimise** *station*
9. **minimise** *market*
10. **minimise** *price*
11. **prefer** *brand* = *A* **or** *brand* = *B* **or** *brand* = *C*
12. *brand* = *A* ≻ *brand* = *B*
13. *brand* = *B* ≻ *brand* = *C*
14. *stars* = 2 **good**
15. **maximise** *stars*

- **if** *uni* > 2.5*Km* **then** *station* ▷ *uni*

Table 3: Running Example Preferences.

Table 4: Available apartments.

Apartment	brand	market	price	stars	station	uni	zone
Ap_A	A	0.45 Km	£100	2	0.3 Km	5.0 Km	2
Ap_B	D	0.40 Km	£115	3	0.6 Km	2.2 Km	1
Ap_C	B	0.20 Km	£95	2	0.3 Km	10.0 Km	3
Ap_D	B	0.60 Km	£105	2	0.5 Km	6.0 Km	2
Ap_E	B	0.30 Km	£100	3	0.5 Km	3.5 Km	2
Ap_F	C	0.40 Km	£125	4	0.9 Km	2.0 Km	1

Definition 4.1. ***Preference Satisfaction Model (PSM)*** *is a partial map from a pair* $\langle option, attribute \rangle$ *to a modifier or its negation,* $PSM : Opt \times Att \mapsto \{\epsilon, \neg\} \times M$*, indicating the most representative modifier that indicates preference for an attribute value of an option.*

Before describing the PSM construction in detail, we introduce auxiliary functions. Each constraint, qualifying or rating preference p is characterised by (i) a modifier, $mod(p)$, e.g. *need*; (ii) a formula, $form(p)$, e.g. $station \leq 1$; and (iii) optionally a condition, $cond(p)$, e.g. $uni \leq 2.5$ (examples are given considering preference 3). As we restrict preferences to refer to a single attribute, $att(p)$ is the attribute that is the referent of the preference, e.g. *station*. An option may or may not satisfy a formula of a constraint or condition, and $sat(formula, o)$ replaces variables from the provided $formula$ with attribute values from option o and evaluates the formula to a boolean value, e.g. $sat(station \leq 1, Ap\_B) = true$. Given

this notation, we define when a preference is applicable to an option.

Definition 4.2. *Preference p is applicable to an option o, $App(p, o)$, if and only if $\nexists c.(c = cond(p)) \vee \exists c.(c = cond(p) \wedge sat(c, o))$.*

Therefore, a preference is applicable to an option, either when it has no condition or when the condition is satisfied by the option. For example, preferences 3, 4 and 5 are applicable only to options Ap_B and Ap_F. Therefore, for each option, there is a subset of monadic preferences that is applicable to it, and each of them is related to a particular attribute. We can thus associate a set of modifiers (or their negation) with each option attribute — $AttMod(MP, o, a)$ — as shown below.

$$
\begin{aligned}
&AttMod(MP, o, a) ::= \\
&\quad \{\langle \epsilon, m\rangle | m = mod(p) \wedge p \in MP \wedge a = att(p) \wedge App(p, o) \wedge sat(form(p), o)\} \\
&\quad \cup \{\langle \neg, m\rangle | m = mod(p) \wedge p \in MP \wedge a = att(p) \wedge App(p, o) \wedge \neg sat(form(p), o)\}
\end{aligned}
$$

In other words, $\langle \epsilon, m\rangle$ is in $AttMod(MP, o, a)$, when there is a preference p in MP that is associated with modifier m (which is p's modifier), is applicable to option o and o satisfies p's formula. If p is applicable to o but o does not satisfy p's formula, $\langle \neg, m\rangle$ is in $AttMod(MP, o, a)$. Note that $AttMod(MP, o, a)$ may be empty. In the case of Ap_F and attribute *station*, for example, we thus have the following attribute modifiers: $AttMod(MP, Ap\_F, station) = \{\langle \epsilon, \textbf{need}\rangle, \langle \neg, \textbf{prefer}\rangle\}$.

Now that we identified all modifiers that indicate preference for a particular attribute of each of the options, we must select the most representative one. Expressive speech acts and rates, widely used by people, have an interpretation that is subjective and specific for each individual. Although they may have different meanings, such as expressing requirement or acceptance, all modifiers also express a degree of preference. Modifiers are categorised as *positive*, indicating a preference for an attribute value; *neutral*, indicating indifference and acceptance for an attribute value; and *negative*, indicating a preference against an attribute value. In addition, modifiers of each category can be stronger in relation to each other. For example, consider the following two preferences: "*I need an apartment whose price is lower than £125*" and "*I prefer an apartment whose price is lower than £100.*" *Need* is stronger in the sense that it tells what *has* to be satisfied. However, when we have two options, the first satisfying only what is needed (e.g. an apartment that costs £110) and the second satisfying what it is needed and preferred (e.g. an apartment that costs £90), *the degree of preference* of the second is stronger than the degree of preference of the first.

Table 5: Modifier scale.

Category	Modifiers	Index
positive	want	9
	love, best	8
	desire, very_good	7
	prefer, like, good	6
	need	5
	require	4
neutral	accept, don't require, don't avoid	3
	neutral, don't love, don't have	2
	don't need, don't desire	1
	$\langle \neg, \mathit{negative\ modifier} \rangle$	0
	$\langle \neg, \mathit{neutral\ modifier} \rangle$	-1
	$\langle \neg, \mathit{positive\ modifier} \rangle$	-2
negative	don't prefer	-4
	avoid	-5
	don't like, bad	-6
	don't want, very_bad	-7
	hate, worst	-8
	don't accept	-9

We adopt a particular ranking that captures this notion to indicate the degree of preference of modifiers, namely *modifier scale*, presented in Table 5, and each subset of modifiers that represents (according to this particular scale) the same degree of preference is associated with an index, which will be used later together with the uncategorised modifiers. The modifier scale is one of the variation points of our technique, as indicated in Figure 1, and it is part of our future work investigate if other rankings or the adoption of an individual-specific interpretation would improve our results.

If more than one monadic preference applies to an option attribute, we use the adopted modifier scale to choose among them as the most representative. There are two possibilities. The first case is when there is at least one monadic preference whose formula is satisfied. According to the modifier scale, from the neutral modifiers (*don't need*, *don't desire*) to the most positive (*want*) there is a stronger preference for an attribute value; and from the neutral modifiers (*accept*, *don't require*, *don't avoid*) to the most negative (*don't accept*) there is a stronger preference against an attribute value. In case there are both positive and negative modifiers, there is inconsistency, which is not taken into account by our technique. Therefore, the strongest modifier from those satisfied (i.e. $\langle \epsilon, m \rangle$) is chosen.

The second case is when there is no monadic preference whose formula is satisfied. Again, there are two possibilities. If there is at least one (unsatisfied) monadic preference whose modifier is positive, the weakest one is chosen. For example, if one option attribute is associated with $\langle \neg, \textbf{prefer} \rangle$ and $\langle \neg, \textbf{require} \rangle$,

Table 6: PSM of the Apartment Decision Problem.

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$
station	$\langle \epsilon, need \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$	$\langle \epsilon, need \rangle$
market						
zone	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \epsilon, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$	$\langle \neg, don't\ accept \rangle$
brand	$\langle \epsilon, prefer \rangle$	$\langle \neg, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$	$\langle \epsilon, prefer \rangle$
stars	$\langle \epsilon, good \rangle$	$\langle \neg, good \rangle$	$\langle \epsilon, good \rangle$	$\langle \epsilon, good \rangle$	$\langle \neg, good \rangle$	$\langle \neg, good \rangle$
price	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$	$\langle \epsilon, require \rangle$

the second is selected, meaning that not even the weakest preference is satisfied. In case there is no monadic preference whose modifier is positive, then the weakest one is chosen (where the weakest is *accept*, *don't require* and *don't avoid*, and the strongest is *don't accept*). This informal explanation to build the PSM is associated with Algorithm 1 that, together with other algorithms of our technique, is presented in Appendix A. The PSM of the presented apartment decision problem built according to the described algorithm is shown in Table 6.

With this model one can see pros and cons against each available option. Even though one of the options might have only positive values, such as *like* and *require*, it does not mean it is the best option, because it may have only minimum acceptable values, and the trade-off with other options might indicate that another option is better. Moreover, other preferences, not processed yet, provide additional information, and this is what we will consider next.

4.1.2. Options-Attribute Preference Model

The second model that will aid us in the decision making process, namely Options-Attribute Preference Model (OAPM), captures comparison relationships between two options, from a perspective of individual attributes. This model shows for which attributes an option is better than or similar to another — or where no conclusion can be reached with the provided preferences. For each OAPM value, which compares an option o to an option o' with respect to an attribute a, there are four possible preference values: (i) +: the attribute value of o is better than o', i.e. $o \succ_a o'$; (ii) −: the attribute value of o is worse than o', i.e. $o' \succ_a o$; (iii) $\sim$: the attribute value of o is as preferred as o', i.e. $o \sim_a o'$; and (iv) ?: no conclusion can be reached. The preference values that associate attribute values of two options are derived from provided user preferences. Besides storing this information, the OAPM also keeps track of the reason that is the cause underlying a preferred value. The OAPM is thus as follows.

Definition 4.3. ***Options-Attribute Preference Model (OAPM)*** *is a total map from* $\langle option_1, option_2, attribute\rangle$ *to a preference value,* $OAPM : Opt \times Opt \times Att \mapsto \{+, -, \sim, ?\} \times Reason$*, indicating which attribute value of these options is the preferred one, and a reason that indicates the (explicit or implicit) type of preference that leads to this conclusion.*

The possible values of *Reason* can be a preference (e.g. maximisation goal), the PSM, or an implicit preference, specifically: *psm*, *max*, *min*, *avpo*, *indiff*, $\langle upper, p\rangle$, $\langle lower, p\rangle$, $\langle around, p\rangle$, and $\langle interval, p\rangle$ (they will be later described). For example, as preference 9 indicates that Ap_B is better than Ap_A with respect to the attribute *market*, the OAPM values are: $OAPM[Ap\_A, Ap\_B, market] = \langle -, min\rangle$ and $OAPM[Ap\_B, Ap\_A, market] = \langle +, min\rangle$ (the reason for the OAPM value is a minimisation goal). The initial OAPM state consists of all values set to ? and therefore, unless there are preferences that allow comparing two attribute values, no conclusion is reached. Note that the OAPM value $OAPM[o_1, o_2]$ is dual to $OAPM[o_2, o_1]$. The specification of our technique sets and uses both OAPM values to make it easier to understand, but its implementation can be optimised by representing just one of the values.

In the pre-processing step, OAPM values are set based on two kinds of information: (i) goal, order and indifference preferences and (ii) the previously built PSM. This information is processed separately in a specific order — PSM, goals, order preferences and indifference preferences — which makes the relationship between two attribute values established by a subsequently processed preference possibly override the current information present in an OAPM value. This is because a user may have general preferences for an attribute, but also have preferences for specific cases, such as stating that an attribute should be minimised and then providing order preferences for specific attribute values. In addition, preferences may refine other preferences, for instance, according to one preference a set of attribute values are considered equally preferred (e.g. preference 11), and specific preferences establish an order among the preferred values (e.g. preferences 12 and 13).

We next describe how the OAPM is constructed, in a declarative way. Presented formulae correspond to rules, which indicate the values to be set in the model. Rules are applied sequentially (following the order in which they are presented) for all pairs of options and attributes, and a subsequent rule may override the values set by a previously applied rule.

PSM. Monadic preferences in isolation do not allow us to compare attribute values, but, with the PSM, these preferences are situated in a context, and we can

conclude that a value that is considered *best* is better than a value that is *good*, for instance. This idea is investigated by Hansson (1990), who discusses the interpretation of "*good*" and "*bad*" in terms of "*better*." Our modifier scale, initially used to select the most representative modifiers, is now used to compare modifiers associated with different options.

Note that the indices presented for categorised modifiers in Table 5 have a gap between negative and neutral modifiers. Besides adjusting indices to give opposite index values to positive and negative modifiers, this gap is used to associate indices with unsatisfied modifiers, i.e. $\langle \neg, m \rangle$. When there is no satisfied modifier for an attribute value, we have three situations: (i) $\langle \neg, positive\ modifier \rangle$, there is one or more positive modifiers to evaluate that attribute value, but it does not satisfy them; (ii) $\langle \neg, neutral\ modifier \rangle$, there is no unsatisfied positive modifier, but there is one neutral that is unsatisfied; (iii) $\langle \neg, negative\ modifier \rangle$, there is no unsatisfied positive or neutral modifier, but the attribute value also does not satisfy a negative one. Situation (iii) is better than the (ii), which is better than (i). When there is a satisfied modifier associated with each of two attribute values being compared — i.e. for both, the PSM value is $\langle \epsilon, modifier \rangle$ — the strongest modifier indicates the preferred value (or equally preferred if both options have the same degree of preference according to the scale), i.e. that with the highest index. We thus use the indices of the modifier scale with additional ones (also shown in Table 5) to compare attributes values.

With these additional indices, we now choose the PSM value associated with the highest index. This way of stating which attribute value is better causes satisfied positive and neutral modifiers to be better than any other unsatisfied one, and satisfied negative modifiers worse than any other unsatisfied one. This interpretation is adopted because users typically explicitly state what they want or do not want, being the absence of preference an indifference (weaker than the explicitly provided indifference), as people usually remember how the experiences felt when they were at their peak (best or worst) (Schwartz, 2005).

Given this approach of establishing a preference relationship between attribute values based on (un)satisfied modifiers, i.e. PSM values, we show the rules used to set the OAPM values, which are applicable only to PSM values that are not null. $PSMIndex(PSM[o, a], scale)$ returns the index of the PSM value according to Table 5. Remember that the OAPM value is composed of a preference value (+, −, ∼, ?) *and* a reason.

$$\begin{aligned} &PSMIndex(PSM[o_1, a], scale) = PSMIndex(PSM[o_2, a], scale) \rightarrow \\ &\quad OAPM[o_1, o_2, a] = \langle \sim, psm \rangle \end{aligned} \tag{1}$$

$$PSMIndex(PSM[o_1, a], scale) > PSMIndex(PSM[o_2, a], scale) \rightarrow \\ OAPM[o_1, o_2, a] = \langle +, psm \rangle \wedge OAPM[o_2, o_1, a] = \langle -, psm \rangle \tag{2}$$

With respect to *station*, Ap_B is thus considered better than Ap_F, as the $PSMIndex$ associated with $\langle \epsilon, prefer \rangle$ (PSM value of Ap_B, *station*) is 6, while the $PSMIndex$ of $\langle \epsilon, need \rangle$ is 5 (PSM value of Ap_F, *station*).

Goals. Goals are restricted to attributes whose domain is ordered, and indicate that an attribute value is considered better than another when its value is higher (maximisation goals) or lower (minimisation goals). Rules shown below set (or change) OAPM values based on goals. They use two additional functions: (i) $type(goal)$, which is *max* for maximisation goals, and *min* for minimisation goals; and (ii) $val(o, a)$, which returns the value of the attribute a of option o.

$$\exists g.(g \in Goal \wedge att(g) = a \wedge App(g, o_1) \wedge App(g, o_2) \\ \wedge val(o_1, a) = val(o_2, a) \rightarrow OAPM[o_1, o_2, a] = \langle \sim, type(g) \rangle) \tag{3}$$

$$\exists g.(g \in Goal \wedge att(g) = a \wedge App(g, o_1) \wedge App(g, o_2) \\ \wedge ((type(g) = max \wedge val(o_1, a) > val(o_2, a)) \\ \vee (type(g) = min \wedge val(o_1, a) < val(o_2, a))) \rightarrow \\ OAPM[o_1, o_2, a] = \langle +, type(g) \rangle \wedge OAPM[o_2, o_1, a] = \langle -, type(g) \rangle) \tag{4}$$

For example, because of preference 9, the OAPM values associated with Ap_C and *market* are set to $\langle +, min \rangle$ with respect to all other options.

Order Preferences. We now proceed to order preferences, which are those that establish an order among two attribute values by explicitly stating the preferred value. Order preferences are transitive, e.g. with preferences 12 and 13 we can derive that brand A is preferred to C. Therefore, from order preferences, we can derive a partial order of attribute values, namely *attribute value partial order (AVPO)*. However, as order preferences may be valid only according to a given condition, two different options may satisfy the conditions of different order preferences. Therefore an AVPO is constructed only with preferences applicable to an option and is specific to a pair of option and attribute. The order preferences of our example have no condition, thus the same partial order (AVPO) is built for all the options with respect to the brand attribute.

An AVPO is a directed acyclic graph $\langle N, A \rangle$, where N is a set of nodes and A is a set of arrows that link nodes. Each node consists of expressions of order preferences, in the form of *attribute* = *value* (e.g., *brand* = *A*), and an arrow

from a node to another represents that the source node is preferred to the sink node. Algorithm 2 (located in Appendix A) shows how an AVPO is constructed for a particular option and attribute. If the output is not a directed acyclic graph, there is a case of inconsistency, which is out of scope. In our example, the order preferences lead to the following AVPO for the *brand* attribute for all options: $brand = A \rightarrow brand = B \rightarrow brand = C$.

An attribute value of an option is preferred to another according to an AVPO, if there is a path from the first to the second, for which we use the notation $ExistsPath(AVPO[o, a], val_1, val_2)$, where val_1 matches a node whose expression is $attribute = val_1$. In order to find such paths, typical graph algorithms are used (Cormen et al., 2001). As different AVPOs may establish different preference relationships, we consider an attribute value of option o_1 better than that of option o_2 if there is a path in the AVPOs of both options, as shown in the next rule.

$$\begin{aligned} &ExistsPath(AVPO[o_1, a], val(o_1, a), val(o_2, a)) \\ &\quad \wedge ExistsPath(AVPO[o_2, a], val(o_1, a), val(o_2, a)) \rightarrow \\ &\quad\quad OAPM[o_1, o_2, a] = \langle +, avpo \rangle \wedge OAPM[o_2, o_1, a] = \langle -, avpo \rangle \end{aligned} \tag{5}$$

All options are associated with the same AVPO with respect to *brand* (presented above), as the same order preferences are applicable to them. According to this AVPO, Ap_A is better than Ap_C, Ap_D, Ap_E and Ap_F, with respect to this attribute.

Indifference Preferences. As opposed to order preferences, indifference is intransitive. A typical example illustrates the reason for this: a person is indifferent to two cups of tea with a difference of 0.1g of sugar in it. If transitivity is adopted, two cups of tea, one with no sugar and another with 1Kg of sugar, by transitivity, are considered equally preferred. Each indifference preference is a set of formulae, establishing indifference for two options' attribute values that satisfy formulae of the same indifference preference, but only if the condition (if any) of the preference is satisfied by both options, as detailed as follows.

$$\begin{aligned} &\exists i.(i \in Indifference \wedge att(i) = a \wedge App(i, o_1) \wedge App(i, o_2) \\ &\quad \wedge \exists f, f'.(f \in form(i) \wedge sat(f, o_1) \wedge f' \in form(i) \wedge sat(f', o_2)) \rightarrow \\ &\quad\quad OAPM[o_1, o_2, a] = \langle \sim, indiff \rangle) \end{aligned} \tag{6}$$

By applying all the OAPM rules to our apartment decision problem, we produce as result the OAPM presented in Table 7.

Table 7: OAPM of the Apartment Decision Problem.

(a) Comparison with Ap_A

	Ap_B	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
market	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle +, psm\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$	$\langle +, avpo\rangle$
stars	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$

(b) Comparison with Ap_B

	Ap_A	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$
station	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$
price	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$

(c) Comparison with Ap_C

	Ap_A	**Ap_B**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
zone	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$

(d) Comparison with Ap_D

	Ap_A	**Ap_B**	**Ap_C**	**Ap_E**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$
market	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$
price	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$

(e) Comparison with Ap_E

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_F**
uni	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle -, psm\rangle$
station	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle \sim, min\rangle$	$\langle +, min\rangle$
market	$\langle +, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, avpo\rangle$
stars	$\langle +, max\rangle$	$\langle \sim, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle -, max\rangle$
price	$\langle \sim, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle +, min\rangle$

(f) Comparison with Ap_F

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**
uni	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$
station	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$
market	$\langle +, min\rangle$	$\langle \sim, min\rangle$	$\langle -, min\rangle$	$\langle +, min\rangle$	$\langle -, min\rangle$
zone	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$	$\langle +, psm\rangle$	$\langle \sim, psm\rangle$	$\langle \sim, psm\rangle$
brand	$\langle -, avpo\rangle$	$\langle +, psm\rangle$	$\langle -, avpo\rangle$	$\langle -, avpo\rangle$	$\langle -, avpo\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$
price	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$	$\langle -, min\rangle$

4.2. Explication

Preferences provided by users always have a literal meaning. For example, the literal meaning of preference 2 of our running example is that apartments that are less than 2.5Km away from the university are preferred to apartments that are farther away than that. However, this sentence can also provide further information: if a maximum desired value is provided, and no minimum value, one can conclude that, considering similar options, lower values are *in general* preferred to higher values. In addition, as this is a soft-constraint, i.e. it can remain unsatisfied if other attributes compensate this loss, the closer an option is to satisfying the preference, the better. In the case of preference 2, it means that between two apartments, both farther away than 2.5Km from the university, the preferred one is the closer.

Preferences that can be derived from other explicit preferences are referred to as *implicit preferences*. We are aware that there may be exceptions, and an implicit preference may be wrongly considered in certain cases — as occurs with

humans. This problem can be tackled with the addition of knowledge specific to an application area, e.g. by stating that usually the higher (or the lower) the value of an ordered attribute is, the better, or by learning what individual users usually mean by their provided preferences. But currently, we adopt a set of implicit preferences that in general derives correct preferences from explicit preferences.

Before introducing our set of implicit preferences, we will describe the context in which they will be considered. Implicit preferences do not override information obtained from explicitly provided preferences, that is, they change OAPM values only when two options are considered similar (w.r.t. an attribute) or no conclusion could be reached. Therefore, the OAPM value for these two options must be either $\sim$ or ? and, if the value is $\sim$, it was not set due to an indifference preference. Moreover, our current implicit preferences are derived from monadic preferences, and as different monadic preferences may be used to derive different implicit preferences, they are derived only when there is only one monadic preference that refers to an attribute and is applicable to a pair of options. Otherwise, we have no information available to choose one.

Our implicit preferences are also valid only for attributes whose domain is ordered. Finally, when keeping track why the OAPM is being updated, the reason is stored in the form of $\langle type, p \rangle$, where *type* is the type of the applied implicit preference, and p is the preference that caused the update. This information is used in the selection step (Section 4.4). Now, we proceed to our implicit preferences.

Around. If a desired value is given for an ordered attribute (a *reference value*), and this preference is not a hard constraint (i.e. it may be left unsatisfied), we infer that the closer the attribute value of an option is to the desired value, the better. Therefore, between two options, the preferred one is that whose value for this attribute has a shorter distance from the reference value.

Interval. Instead of providing a single desired attribute value, users may provide an *interval*. In these cases, the provided monadic preference is associated with a formula that is an instance of $attribute > lowerBound$ **and** $attribute < upperBound$, or $\geq$ and $\leq$, instead of $>$ and $<$. The attribute value of an option that has the lower distance from the interval than another as preferred; or the opposite, if the preference modifier is negative.

Upper Bound. In the upper bound case, an upper limit is provided for an attribute (as in preference 2), which is given in a monadic preference whose formula is an instance of $attribute < value$ or $attribute \leq value$. Due to this upper bound, we infer that there is a minimisation goal for this attribute. Note that

Table 8: Updated OAPM of the Apartment Decision Problem.

(a) Comparison with Ap_A

	Ap_B	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, psm\rangle$	$\langle +, upper\rangle$	$\langle +, upper\rangle$	$\langle -, upper\rangle$	$\langle -, psm\rangle$
zone	$\langle -, lower\rangle$	$\langle +, psm\rangle$	$\langle \sim, lower\rangle$	$\langle \sim, lower\rangle$	$\langle -, lower\rangle$
stars	$\langle -, max\rangle$	$\langle \sim, around\rangle$	$\langle \sim, around\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$

(b) Comparison with Ap_B

	Ap_A	**Ap_C**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle -, upper\rangle$
zone	$\langle +, lower\rangle$	$\langle +, psm\rangle$	$\langle +, lower\rangle$	$\langle +, lower\rangle$	$\langle \sim, lower\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle \sim, around\rangle$	$\langle -, max\rangle$

(c) Comparison with Ap_C

	Ap_A	**Ap_B**	**Ap_D**	**Ap_E**	**Ap_F**
uni	$\langle -, upper\rangle$	$\langle -, psm\rangle$	$\langle -, upper\rangle$	$\langle -, upper\rangle$	$\langle -, psm\rangle$
zone	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$	$\langle -, psm\rangle$
stars	$\langle \sim, around\rangle$	$\langle -, max\rangle$	$\langle \sim, max\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$

(d) Comparison with Ap_D

	Ap_A	**Ap_B**	**Ap_C**	**Ap_E**	**Ap_F**
uni	$\langle -, upper\rangle$	$\langle -, psm\rangle$	$\langle +, upper\rangle$	$\langle -, upper\rangle$	$\langle -, psm\rangle$
zone	$\langle \sim, lower\rangle$	$\langle -, lower\rangle$	$\langle +, psm\rangle$	$\langle \sim, lower\rangle$	$\langle -, lower\rangle$
stars	$\langle \sim, around\rangle$	$\langle -, max\rangle$	$\langle \sim, around\rangle$	$\langle -, max\rangle$	$\langle -, max\rangle$

(e) Comparison with Ap_E

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_F**
uni	$\langle +, upper\rangle$	$\langle -, psm\rangle$	$\langle +, upper\rangle$	$\langle +, upper\rangle$	$\langle -, psm\rangle$
zone	$\langle \sim, lower\rangle$	$\langle -, lower\rangle$	$\langle +, psm\rangle$	$\langle \sim, lower\rangle$	$\langle -, lower\rangle$
stars	$\langle +, max\rangle$	$\langle \sim, around\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle -, max\rangle$

(f) Comparison with Ap_F

	Ap_A	**Ap_B**	**Ap_C**	**Ap_D**	**Ap_E**
uni	$\langle +, psm\rangle$	$\langle +, upper\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$	$\langle +, psm\rangle$
zone	$\langle +, lower\rangle$	$\langle \sim, lower\rangle$	$\langle +, psm\rangle$	$\langle +, lower\rangle$	$\langle +, lower\rangle$
stars	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$	$\langle +, max\rangle$

monadic preferences may be associated with negative modifiers, and in the case the inference is the opposite: there is a maximisation goal.

Lower Bound. As opposed to the previous case, a lower bound can be provided for an attribute, with monadic preferences whose formula are instances of *attribute* $>$ *value* or *attribute* $\geq$ *value*, thus indicating that the goal is to maximise the value of this attribute (or to minimise it if the preference modifier is negative).

There are three cases to which implicit preferences are applicable in our running example, which are related to the preferences 1, 2 and 14. Now, the relationship between options with respect to the attribute *zone* is established by the goal of minimising the value of this attribute (upper bound preference, with a negative modifier), and the attribute *uni*, which has also the goal of minimising the value of this attribute (upper bound preference, with a positive modifier). The unique modifier preference with respect to *stars* indicates that the closer that the apartment starts are to 2, the better (around preference), but option attribute values are either already decided by the explicitly provided goal, or equal, thus the comparison remains $\sim$. Table 8 shows the updated OAPM, only with attributes that have changed. The OAPM reason of implicit preferences include both a type of preference and an associated preference, but we omit the latter for simplicity.

After executing the steps for building our two computational models that support the decision making process, and updating them by considering implicit pref-

erences, we still have preferences provided by users that we have not taken into account, which are don't care preferences and priorities over preferences and over attributes. These will be used later, when resolving trade-off situations for choosing an option but, before that, we use the constructed PSM and the OAPM to eliminate options, as presented next.

4.3. Elimination

Before analysing option pros and cons to select one option, we can eliminate those that are clearly worse than others. This is the case of: (i) dominated options (Section 4.3.1); and (ii) options that do not satisfy hard constraints (Section 4.3.2).

4.3.1. Eliminating Dominated Options

We begin eliminating options that, for at least one attribute, are worse than another option, and are not better than it for the remaining attributes. In this situation, we say that the options to be discarded are dominated by another. Based on the information provided by the OAPM, which was constructed based on provided user preferences, we can define *domination* as a binary relation that indicates when an option dominates another, formally presented in Definition 4.4.

Definition 4.4. *Let $o_1, o_2 \in Opt$ be two options, and $a \in Att$ an attribute. It is said o_1 dominates o_2 — $dominates(o_1, o_2)$ — when $\exists a.(OAPM[o_1, o_2, a] = +) \wedge \forall a.(OAPM[o_1, o_2, a] \neq -)$.*

For the domination relation be true, o_1 must have an attribute value that is preferred to the attribute value of o_2. In addition, for all other attribute values, o_1 has to be at least as good as o_2. Therefore, we have a reason to reject o_2, and there is no other positive aspect of this option that can balance its negative aspects, w.r.t. o_1. Domination also holds in the absence of information about the attribute comparison of two options ($OAPM[o_1, o_2, a] =?$), if there is at least one attribute whose OAPM value is +.

Based on the definition of domination, we now define the set of dominated options, *Dominated*. All options dominated by at least one other option are in the *Dominated* set and are discarded, i.e. $dominates(o_1, o_2) \rightarrow o_2 \in Dominated$.

In Tables 7(a) and 7(e), it can be seen that the OAPM has the value + or $\sim$, for all attributes of options Ap_A and Ap_E, when compared to Ap_D. Therefore, it can be said that $dominates(Ap\_A, Ap\_D)$ and $dominates(Ap\_E, Ap\_D)$, thus $Ap\_D \in Dominated$.

4.3.2. Applying Cut-off Values

The second set of eliminated options is composed of options that do not satisfy hard constraints of users. We consider hard constraints preferences that are either qualifying or rating statements with one of these four modifiers: (i) *don't accept*; (ii) *hate*; (iii) *require*; and (iv) *need*.

Other modifiers, such as *very good*, *want* and *very bad*, are also strong preferences from users, but they are not considered at this moment, because options have positive and negative aspects w.r.t. each other (otherwise it is a case of domination) and, even though an option has an attribute value that is *very bad*, it may be amortised by other positive aspects of this option. This argument is not valid for the four modifiers considered as hard constraints, because they are interpreted as *non-compensatory* attribute values, i.e. those that cannot be compensated with any benefit provided by other attributes.

In order to identify the options that are discarded due to cut-off values, we use the information captured by the PSM. First, we show how to select options associated with hard negative modifiers, and then those associated with unsatisfied hard positive modifiers. In the first case, the options selected to be part of the *CutOff* set, i.e. the set of options discarded due to a cut-off value, are those that have at least one attribute associated with $\langle \epsilon, \mathit{don't\ accept} \rangle$ or $\langle \epsilon, \mathit{hate} \rangle$ in the PSM. By construction, every option that satisfies a preference with either the "*don't accept*" or the "*hate*" modifier has the PSM value evaluated for these modifiers for the respective attribute. Therefore, these constraints are always respected.

Differently from the negative modifiers above, *require* and *need* are considered hard constraints unless another positive experience is provided. For example, assume these two preferences: "*I require an apartment at zone 1*", and "*I accept one in zone 2*." In this case, the second preference changes the first preference to a soft preference, as it indicates an exception to the requirements. So, even though an apartment in zone 2 does not satisfy a requirement, it is not eliminated due to a cut-off. Therefore, *require* and *need* are usually hard constraints, but users may add acceptable exceptions, as in this example. So, we exclude options that satisfy neither a requirement or need, nor any other monadic preference whose modifier is neutral or positive, w.r.t. a particular attribute. However, if an option does not satisfy the requirement or need, but satisfies a monadic preference whose modifier is negative, then it is discarded when there is at least one option that satisfies the requirement or need. This interpretation is adopted because if users provide other preferences related to an attribute besides a requirement or need, it is an indication that they are a soft constraint, but still have a strong preference for their satisfac-

tion. By construction, if no monadic preference is satisfied by an option, and one of them is associated with a requirement (or need), the PSM value of this option will indicate an unsatisfied requirement (or need).

The PSM presented in Table 6 shows that $Ap\_C \in CutOff$, as $PSM[Ap\_C, zone] = \langle \epsilon, don't\ accept \rangle$. Note that an option eliminated because of domination may also be discarded due to a cut-off value, i.e. it is possible that $Dominated \cap CutOff \neq \emptyset$.

4.4. Selection

After performing the elimination step of our process, our set of available options is now reduced to a subset of options (which we refer to as *Acceptable*, but it is also referred to as *consideration set* in the literature), which requires us to resolve trade-offs to make a choice. Our technique, inspired by human decision making, analyses costs and benefits of options and additional factors that humans typically adopt while making decisions. Our goal is to understand how people resolve trade-offs, when demanding adequate time and effort to make a decision and this is the reasoning process we adopt to make automated choices. Avoiding extreme options (best for some attributes and worst for others) and analysing the trade-off contrast (influenced by other options), are the two principles that humans adopt that our technique incorporates (Simonson and Tversky, 1992).

The following sections describe the selection step of our technique in four parts. First, we describe how the benefits and costs of each option in the *Acceptable* set, where $Acceptable = Options \backslash (Dominated \cup CutOff)$, are evaluated with respect to each individual attribute. Then, we show how we assess the overall benefits and costs of options with respect to each other based on the previous evaluation of each individual attribute. Later, we consider the two main principles that are typically adopted by people when making decisions. Finally, we show how all these evaluations are put together and used to make a final decision.

4.4.1. Cost-benefit Analysis

The first part of the process of selecting an option consists of evaluating each pair of options and assessing their relative benefits and costs, using the information provided by the OAPM. The costs of option o_1 w.r.t. o_2 are the benefits of option o_2 w.r.t. o_1, and vice-versa. We compute the cost of o_1 compared to o_2 w.r.t. an attribute a to a real value ranging from 0 to 1, captured by a function we build: $AttCost : Opt \times Opt \times Att \rightarrow \{c | c \in \mathbb{R} \wedge 0 \leq c \leq 1\}$.

This value indicates *how much* one option is better than another, w.r.t. to each attribute, which will be used to evaluate the overall option costs. In essence, our cost function transforms qualitative information into quantitative values. These

quantitative values comprise a function similar to a utility function (UF) (Keeney and Raiffa, 1976), as it is a weighted sum of values given for individual attributes, which represent how much an individual prefers each attribute value. Our function and UFs have, however, two main differences. First, the cost function consists of *differences* between values as the cost is obtained by means of a *pairwise* comparison between options (as people usually do), and uses the *specific preferences* for each option pair, as there are many types of preferences applicable for pairs of options (note that using qualitative preferences to quantitatively evaluate preference for options is also a challenge). Second, the way we obtain these cost values is novel: we exploit natural-language-like expressions instead of submitting users to iterative processes, which demand high cognitive effort and time from users. Furthermore, this function is not decisive for making a choice: as introduced before, we will also add two factors that influence preferences for options, which are associated with two principles of human decision making. Therefore, option costs are not the unique factors that are needed for making the decision.

The attribute cost is 0, if the $OAPM[o_1, o_2, a] \neq \langle -, r \rangle$; otherwise, i.e. if $OAPM[o_1, o_2, a] = \langle -, r \rangle$, then the reason r is used to compute the $AttCost[o_1, o_2, a]$. We next describe this computation for the seven possible reasons r: PSM, order preferences, goal, lower bound, upper bound, around and interval. The remaining one, indifference, always causes OAPM values to be set to $\sim$, and the attribute cost is also 0.

PSM. Our modifier scale (Table 5) allows us to identify whether a modifier is stronger than another, and consequently the preference for a value compared to another when different modifiers are used to qualify these values. However, as in this step we aim to asses *how much* one attribute value is preferred to another, we have to go beyond the order given by this scale. In order to make this assessment, we associate a numeric value with the index of each PSM value, and therefore an option cost for the particular attribute with respect to another is calculated based on the difference between the values associated with the PSM values.

The association of a numeric value with each PSM value is given by a function that we refer to as f_m, which corresponds to a variation point of our technique. We have considered three different functions (linear, quadratic, and logarithmic) for generating values for modifiers. These values are normalised to values between 0 and 1, considering the possible modifiers (or their negation) that can be associated with any of the two options being compared. Therefore, for all monadic preferences that either $App(p, o_1)$ or $App(p, o_2)$, we select the $\langle \epsilon, m \rangle$ or $\langle \neg, m \rangle$ that has the maximum and minimum indices, i.e. we obtain the maximum and minimum

values associated with a PSM value that these options can have.

We have adopted the **log** function ($f_{m_{lg}}$) in our approach, chosen based on experimentation. This function makes the difference between strong modifiers, such as *want* and $best$, smaller than the differences between modifiers in the middle of the scale, such as *neutral* and *don't avoid*, and therefore the preference is stronger when we compare positive modifiers with negative modifiers. For example, the cost of values rated with *neutral* compared to values qualified with *don't prefer* is higher than the cost of values qualified with *want* compared to values qualified with $desire$, even though for both the index differences is of two units.

When calculating the cost of the attribute *uni* with respect to options Ap_A and Ap_B, the maximum and minimum possible values for these options are those associated with $\langle \epsilon, prefer \rangle$ and $\langle \neg, prefer \rangle$, respectively, which are also the PSM values associated with Ap_B and Ap_A. Therefore, we have $|f_{m_{lg}}(6) - f_{m_{lg}}(-2)|/(f_{m_{lg}}(6) - f_{m_{lg}}(-2))$, which is 1.0 — the value of $AttCost(Ap\_A, Ap\_B, uni)$.

Order Preferences. Attribute value partial orders (AVPOs) allow comparison of attribute values and identifying the preferred one; however, as in our modifier scale, it is not possible to know how much one value is preferred to another. So, in order to obtain this information, we also associate numeric values with AVPO nodes, and for that we use information from the monadic preferences. Each AVPO is associated with an option and an attribute, and the monadic preferences considered are those that have their condition satisfied by that option and attribute. We initiate this process by tagging AVPO nodes with a modifier from the monadic preferences whose formula is satisfied by the domain value associated with the node: for selecting among multiple modifiers, we follow the same rules used for building the PSM, but here we have only satisfied modifiers. Based on this tagging, the AVPO nodes are associated with a numerical value, as summarised in Figure 2 and explained in detail next.

Extreme Nodes. To guarantee the existence of tagged values in the AVPO, we tag all most preferred values and all least preferred nodes (fourth column of Figure 2). The former is tagged with *want* (in our current modifier scale, it is the strongest modifier) and the latter with *neutral*. If there are values in the partial order tagged with a stronger modifier than *want* or *neutral*, for instance $A \succ B \succ C$, and B is tagged with *avoid*, the extreme untagged nodes receive these tags for keeping consistency, i.e. A is tagged with *want* (as the default) and C with *avoid*. We tag least preferred nodes with *neutral* by default, because people typically provide an order for preferred or acceptable values, and do not mention not preferred ones — we confirmed this in our previous study (Nunes et al., 2010).

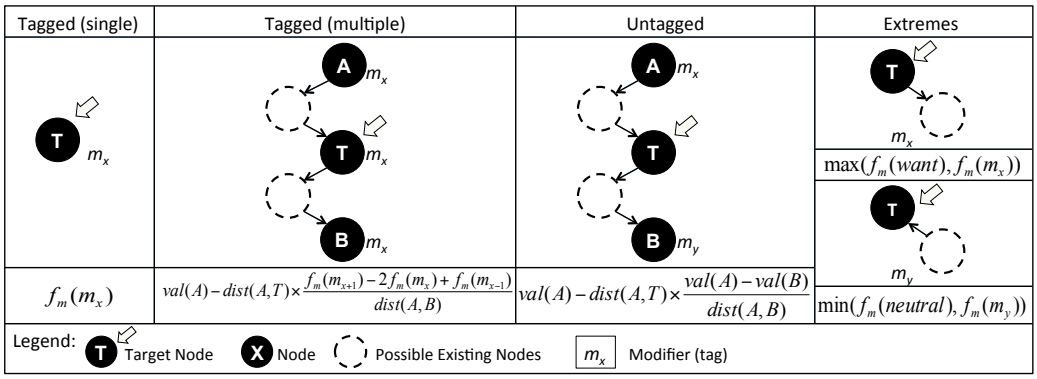

Figure 2: Calculating node values.

Tagged Nodes. If the node is tagged with a modifier (first and second columns of Figure 2), it receives the value according to the values given for the modifier scale. However, there may be different values tagged with the same modifier and ordered in a sequential way, such as in the running example in which all nodes of the AVPO are tagged with the $prefer$ modifier.

In order to address this issue, we first search for the most distant node that has the same modifier of the target modifier, either searching through the parents or children of nodes. This is done by Algorithm 3, shown in Appendix A. Then, two situations can happen, handled by Algorithm 4: $TaggedNodeValue(node)$. If the node tagged with a particular modifier is unique, its node value is given by the f_m function. In the second situation — more than one node is tagged with the same modifier — the most preferred value (brand *A*, in our running example) is associated with the numeric value related to the modifier, plus half of the difference between the modifier value and the modifiers whose index is increased in one unit, according the modifier scale (in the example, *desire*). Similarly, the least preferred value (brand *C*) is associated with the value related to the modifier, minus half of the difference between the modifier value and the modifiers whose index is decreased in one unit (in the example, *need*). With these values, we divide the difference between the values associated with the most and least preferred values by their distance (which is two, in the case of brands *A* and *C*) for obtaining the difference between any two values, which we refer to as $step$, and with it we are able to calculate the value of the remaining nodes. Therefore, the value associated with *B* is the value of *A* minus the distance between *A* and *B* times the $step$.

Untagged Nodes. If the node is not tagged with a modifier (third column of Figure 2), we first obtain the maximum and minimum surrounding node values

of the target node (which is done through the execution of Algorithm 5). We find the closest tagged nodes that are preferred than the target node (maximum), and the closest tagged nodes less preferred than the target node (minimum). Then, we choose from these tagged nodes by selecting those that have the smaller difference from the target node: we consider their unsigned numerical value (as they are tagged, it is calculated in the way we explained above) and divide by the distance between them and the target value (*step*). Next, we calculate the numerical value for the node immediately above (or below) the target node: we take the numerical value of the tagged node and subtract (or add) from it the step times the distance from the tagged node to the target node minus one, as we are calculating the value of the node immediately above or below. The lower (higher) numerical value calculated for this node is chosen, and therefore we guarantee that all more preferred nodes (than the target node) are associated with a higher numerical value and all less preferred nodes (than the target node) are associated with a lower numerical value. After choosing the preferred and less preferred nodes, we calculate the difference between their numerical values and divide it by their distance (*step*), and then we calculate the numerical value related to the target node as above — as shown in Algorithm 6: $UntaggedNodeValue(node)$.

After associating tags and numerical values with the AVPO nodes, we can calculate the costs of an option with respect to an attribute using this information. The cost associated with the attribute values of two options according to a given AVPO is shown in Equation 7, where $Node(AVPO[o, a], o_1)$ gives the node of the AVPO of option o and attribute a that is associated with the attribute value a of o_1. The cost is normalised in a similar manner as with the PSM. We use as maximum and minimum values those associated with indices of modifiers and their negation that tag any of the AVPO nodes. Finally, as there are two AVPOs, one associated with each option, the final cost is the average of their respective costs.

$$AVPOAttCost(o, o_1, o_2, a) = \frac{|NodeVal(Node(AVPO[o, a], o_1)) - NodeVal(Node(AVPO[o, a], o_2))|}{f_m(maxIdx(AVPO[o, a])) - f_m(minIdx(AVPO[o, a]))} \quad (7)$$

Goal, Lower Bound and Upper Bound. In the case that an option is considered worse than another (with respect to an attribute) due to a goal, upper or lower bound, we again exploit monadic preferences. Each attribute is associated with a domain, and we tag different domain values of attributes associated with goals (or lower and upper bound preferences) with values of modifiers of monadic pref-

erences satisfied by the domain values. The tagging, according to the formula of monadic preferences, is as follows.

(i) *attribute* = *value*: the domain value *value* is tagged with the value associated with the preference modifier.

(ii) *attribute* > $value_1$ **and** *attribute* < $value_2$: the domain values $value_1$ and $value_2$ are tagged with the preference modifier, plus and minus the difference from this modifier to the closest modifiers (as with AVPO nodes tagged with the same modifier). Note that it is possible to have ≥ and ≤ instead of > and <, respectively.

(iii) Domain boundaries, if not tagged, are tagged with the numeric values associated with *don't want* (minimum value), which is the minimum modifier that is not a hard constraint, and *want* (maximum value), in case of maximisation goal (or lower bound). If the goal is a minimisation (or upper bound), the boundaries tags are inverted. We do not use preferences with *don't accept* and *hate*, because these modifiers are hard constraints, but as in AVPOs, they are used to tag domain values only to keep consistency if there are monadic preferences that use them.

With this tagging, we keep the maximisation and minimisation goals and associate a degree of preference (*DoP*) with specific domain values. Now, we use this degree of preferences to measure attribute costs. As each domain value is now surrounded by two tagged values (or it is a tagged value), we are able to derive a degree of preference for all domain values, and the cost is the difference between them, normalised according to the maximum and minimum degrees of preference, which are given by the domain boundaries. Given an attribute value y_a, whose closest tagged attribute values are x_a, tagged with t_x, and z_a, tagged with t_z, we can calculate the parameters a and b of a linear function $DoP(x) = ax + b$, where $a = (t_x - t_z)/(x_a - z_a)$, and $b = t_x - ax_a$, and then calculate the degree of preference of y_a. If y_a is tagged, it already has an associated degree of preference. Although goals can be either of maximisation or of minimisation, we use the same form of calculating the attribute cost, as the difference is the same in both cases, and the cost is associated only with the option whose OAPM value is −.

Around. For assessing the cost using an around preference, we make a similar calculation as above, but many modifiers are not helpful in this case, as there is solely one monadic preference that is applicable to the options being compared — this is a requirement to apply the around implicit preference. We evaluate the

Table 9: Cost-benefit Analysis for the Apartment Decision Problem.

(a) $AttCost(Ap\_A, o', a)$

	w_i	**Ap_B**	**Ap_E**	**Ap_F**
uni	0.179	1.000	0.100	1.000
station	0.196			
market	0.132	0.071	0.214	0.071
zone	0.210	0.200		0.200
brand	0.094			
stars	0.030	0.027	0.027	0.054
price	0.158			
Cost		0.231	0.047	0.232

(b) $AttCost(Ap\_B, o', a)$

	w_i	**Ap_A**	**Ap_E**	**Ap_F**
uni	0.196			0.013
station	0.179	0.250	0.083	
market	0.132		0.143	
zone	0.210			
brand	0.094	1.000	1.000	1.000
stars	0.030			0.027
price	0.158	0.500	0.500	
Cost		0.218	0.207	0.098

(c) $AttCost(Ap\_E, o', a)$

	w_i	**Ap_A**	**Ap_B**	**Ap_F**
uni	0.179		1.000	1.000
station	0.196	0.166		
market	0.132			
zone	0.210		0.200	0.200
brand	0.094	0.018		
stars	0.030			0.027
price	0.158			
Cost		0.034	0.221	0.222

(d) $AttCost(Ap\_F, o', a)$

	w_i	**Ap_A**	**Ap_B**	**Ap_E**
uni	0.196			
station	0.179	0.500	0.250	0.333
market	0.132			0.143
zone	0.210			
brand	0.094	0.036		0.018
stars	0.030			
price	0.158	0.833	0.333	0.833
Cost		0.225	0.098	0.212

attribute cost based on the difference of between attribute values and the reference value, which ranges from 0 (the attribute value is equal to the reference value, thus the distance from this value is 0) to the longest distance from the reference value, considering the attribute domain. As here we cannot use the difference between modifiers (as there is only one modifier, associated with the reference value), we use a function $f_d(dist)$ (currently instantiated as a linear function), to evaluate cost in terms of the distance from the reference value.

Interval. Similarly to the around preference case, we assess the cost using an interval preference as a basis using the distance from a reference value, which is now an interval. The range of possible distances is from 0 (the attribute value is in the interval) to the longest distance from the interval extremes (considering the attribute domain), which are used by f_d to calculate cost.

Given these different ways of calculating the attribute cost $AttCost(o_1, o_2, a)$ based on the reasons for establishing a preference between attribute values of two options, we show the attribute costs for our running example in Table 9.

Overall Option Costs. Up to now, we have considered the costs of an option with respect to another by considering attributes in isolation, and now we look at the overall option costs (also w.r.t. option). This is performed by taking into account

the priorities provided — which can be preference priority, attribute priority and attribute indifference — and building an attribute partial order (*attPO*) for each option, as different priorities can be applicable to different options.

As preferences, priorities *pri* also have a condition *cond*(*pri*), and we present a similar applicability definition.

Definition 4.5. *Priority pri is applicable to an option o, App*(pri, o)*, if and only if* $\nexists c.(c = cond(pri)) \vee \exists c.(c = cond(pri) \wedge sat(c, o))$.

We initially take into consideration preference priorities (applicable to a particular option), which associates a number with preferences, meaning that the lower the number associated with the preference is, the more important it is. Each preference is related to a single attribute (according to our assumptions), and therefore the attribute order follows the order implied by the numbers associated with the preference. Because there may be many preferences associated with an attribute, we consider the lowest number. This is done by Algorithm 7.

Attribute priority and attribute indifference modify the order given by preference priorities. In the first case, if the attribute priority is inconsistent with the order of preference priorities, the attribute that was, before, considered less important becomes the attribute immediately more important than the other attribute referred in the given attribute priority. In the second case, if the attribute indifference is inconsistent with the order of preference priorities, the least important attribute becomes as important as the previously more important attribute. Algorithms 8 and 9 show how this swapping process is performed due to attribute priority and attribute indifference, respectively. Finally, attributes associated with a *don't care* preference are excluded from attribute partial order. It is important to highlight that, as we assume consistency, priorities do not form a cycle. In our running example, for option Ap_A , preference priorities result initially in the following order: *zone* $\succ$ **uni** $\succ$ **station** $\succ$ *price* $\succ$ *market* $\succ$ *brand* $\succ$ *stars*. By considering the given attribute priority, *station* is swapped with *uni*: *zone* $\succ$ **station** $\succ$ **uni** $\succ$ *price* $\succ$ *market* $\succ$ *brand* $\succ$ *stars*.

Given the attribute order, we consider the least important attributes as having level 1 in the order, and the longest path in the order from the least important attributes to the most important ones is referred to as *length*(*attPO*). Then, we use a logarithmic function ($f_a(x) = \alpha \log x + \beta$) for calculating the attribute weights when considering the overall option benefits. We establish the following points for the function: $f_a(1) = 1$, and $f_a(length(attPO)) = length(attPO)$. The first point indicates that attributes in the first level of the order have the minimum weight, which is 1, and the second point shows that attributes in the last level have

the maximum weight, which is *length*(*attPO*). The logarithmic function, with the characteristics imposed by the points we established, gives a much higher priority to more important attributes, and these more important attributes have a smaller difference among them (in comparison with a linear function). This is a default form we are adopting for calculating attribute weights, which was also selected based on experimentation, and is a variable part of our technique. The parameters α and β of the logarithmic function for a particular level of attributes are $\alpha = (length(attPO) - 1)/(\log length(attPO))$ and $\beta = 1$, respectively.

Based on the logarithmic function with the calculated parameters, the weight of each attribute $a_i \in Att$ is as shown below.

$$w_i = \frac{f_a(level(a_i))}{\sum_{a_j \in Att} f_a(level(a_j))}$$

Finally, now that we have the costs of an option o_1 with respect to o_2, for each individual attribute, and we also have the attributes weights, we calculate the overall benefits from o_1 with respect to o_2 using an weighted sum, as presented next. This function, which denotes the costs of all options w.r.t. each other option, calculated for our running example is shown in the last row of Table 9, which also details the attribute weights for each option.

$$Cost(o_1, o_2) = \sum_{a_i \in Att} w_i \times AttCost(o_1, o_2, a_i)$$

4.4.2. Trade-off Contrast

The result of not having dominated options in the set of acceptable options is that for any two options, one option is better for one or more attributes and the same applies to the other. As a consequence, a trade-off must be resolved for choosing one of the two options. According to Simonson and Tversky (1992), when people make choices they do not look only for the two options being compared, but analyse the cost-benefit relationship between two options compared with the cost-benefit relationship between all other options. This reasoning of comparing the trade-offs of the whole set of options is referred to as *trade-off contrast*, and is not in accordance with traditional decision making theory, because in the latter case the preference of a decision maker for two options is independent of the other available options, referred to as the axiom of *context invariance*.

Therefore, we incorporate a new factor in the process of choosing an option, which is captured by a function that shows the trade-off between two options: $to : Opt \times Opt \rightarrow \mathbb{R}$. We build the trade-off (*to*) as a partial function whose

domain is every pair of options that satisfies $Cost(o_1, o_2) < Cost(o_2, o_1)$ and is associated with the options' cost-benefit relationship: $Cost(o_1, o_2)/Cost(o_2, o_1)$. Because $Cost(o_1, o_2) < Cost(o_2, o_1)$, *to* is always value in the interval $[0, 1]$ and $Cost(o_2, o_1)$ cannot be 0. The average of all values of *to* is referred to as avg_{to}.

The trade-off between two options does not have a meaning in an isolated manner; when we have only two options, all we know is that one option has higher or lower cost than another. When there are other options, and the decision maker observes that the cost-benefit relationship is better for other options, this is seen as a negative aspect of the option and the benefits become smaller. That is, the option requires giving too much for receiving just a little in exchange.

Given the structure we build to store trade-offs, *to*, we now calculate the option costs with respect to trade-off, having as a basis the average of the trade-off between a particular option with the others (which is represented by the average of trade-offs $avg_{to}(o)$) and the trade-off among all options (which is represented by the average of all trade-offs avg_{to}). If $Cost(o_1, o_2) < Cost(o_2, o_1)$ and the trade-off relationship of o_1 is higher (i.e. worse) than avg_{to}, then we have one more cost of o_1 w.r.t. o_2. If the trade-off is lower (i.e. better) than the average, than it is counted as a benefit, and therefore as a cost for o_2. The function $ToContrast(o_1, o_2)$, which captures this notion of trade-off contrast, is shown below.

$$ToContrast(o_1, o_2) = \begin{cases} avg_{to}(o_1) - avg_{to} & \text{if } to(o_1, o_2) \text{ is defined} \\ & \text{and } avg_{to}(o_1) > avg_{to} \\ avg_{to} - avg_{to}(o_1) & \text{if } to(o_2, o_1) \text{ is defined} \\ & \text{and } avg_{to}(o_2) < avg_{to} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

4.4.3. Extremeness Aversion

Another aspect that people take into consideration when making a decision is how extreme options are. Extreme options are those that have a large improvement for one attribute (or set of), e.g. quality, and a high penalisation for another attribute (or set of), e.g. price. In general, people avoid extreme options (Simonson and Tversky, 1992), and this is referred to as *extremeness aversion*.

In order to evaluate how extreme options are, we compare option attribute values to the best possible values, measuring the distance between them. As best values are subjective to each individual, we use the preferences applicable to each option to identify the best value for each attribute. As these values are better or equal to the particular option being analysed o, each attribute value of o can be associated with an attribute cost, which ranges from 0, i.e. the attribute value is

equal to the best value, to 1, i.e. the attribute value is the worst possible value. Each of these costs is referred to as distance from best, or $bestDist(o, a)$.

The procedure is similar to making a cost-benefit analysis of the option being analysed with a hypothetic option whose attribute values are the best. Preferences to identify best values are processed in the inverse order of that used to build the OAPM, consequently we will keep the same precedence order, as earlier processed preferences may have their OAPM value overridden. Best values are identified in the following way, when attribute a of option o is being analysed. If there is a *don't care* preference associated with a (and is applicable to o), a is not taken into account. Given this way of identifying best values, we calculate $bestDist(o, a)$ in the same way that attribute costs are calculated, but comparing options with best values instead of other options.

An extreme option has low costs for some attributes ($bestDist(o, a)$ close to 0) and high costs for others ($bestDist(o, a)$ close to 1), therefore we evaluate how extreme the option is by calculating the standard deviation of the function $bestDist$ for a particular option, for all attributes, which is a value between 0 and 1.

$$ext(o) = STDEV(\{bestDist(o, a_i) | i = 1...|Att|\})$$

Finally, as the more extreme the option is, the more people avoid it, it is considered that the more extreme option, between two options, has a cost with respect to the other option. So, in order to capture this aspect, we define $ExtAversion : Options \times Options \rightarrow \mathbb{R}$, which represents the cost of the first option compared to the second, with respect to the extremeness aversion principle. This function, presented below, shows how the extremeness aversion is calculated: the more extreme option has a cost that is the difference between the extremeness values of the two options, and, as the less extreme option has no cost with respect to the other, the value is 0.

$$ExtAversion(o_1, o_2) = \begin{cases} ext(o_1) - ext(o_2) & \text{if } ext(o_1) > ext(o_2) \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

4.4.4. The Decision Function: Comparing Relative Option Values

After executing the previous steps, we have analysed three aspects when comparing options: their costs, the trade-off relative to the set of available options, and how extreme they are. The last two aspects are also seen as costs (or benefits): if the trade-off is higher than the average, it is also considered as a cost, and a more extreme option has a cost when compared to a less extreme. So the final value of an option with respect to another combine these three aspects in a

Table 10: Decision Function of the Apartment Decision Problem.

(a) Ap_A

	B	**E**	**F**
Cost	0.231	0.047	0.232
ToConstrast	0.000	0.191	0.000
ExtAversion	0.017	0.015	0.000
d	0.141	0.078	0.139

(b) Ap_B

	A	**E**	**F**
Cost	0.218	0.207	0.098
ToConstrast	0.017	0.017	0.000
ExtAversion	0.000	0.000	0.000
d	0.135	0.129	0.059

(c) Ap_E

	A	**B**	**F**
Cost	0.034	0.221	0.222
ToConstrast	0.000	0.000	0.000
ExtAversion	0.000	0.002	0.000
d	0.021	0.133	0.133

(d) Ap_F

	A	**B**	**E**
Cost	0.225	0.098	0.212
ToConstrast	0.052	0.052	0.052
ExtAversion	0.042	0.060	0.057
d	0.154	0.080	0.149

weighted sum of these costs, which can be seen in Equation 10, comprising our *decision function* $d(o_1, o_2)$. We are currently considering default weights (the last variation point of our technique), which are 0.25 for trade-off contrast and 0.15 for extremeness aversion. Based on $d(o_1, o_2)$, we identify the chosen option, which is the option that has less or equal disadvantages ($d(o_1, o_2) \leq d(o_2, o_1)$) than every other option of the *Acceptable* set, i.e. those that are better or equal to the other options. If different options have the same decision value with respect to another ($d(o_1, o_2) = d(o_2, o_1)$), and they are better than every other option, we randomly choose one of them.

$$\begin{aligned} d(o_1, o_2) = (1 - w_{to} - w_{ea}) \times Cost(o_1, o_2) \\ + w_{to} \times ToContrast(o_1, o_2) + w_{ea} \times ExtAversion(o_1, o_2) \end{aligned} \tag{10}$$

Considering our apartment example, if the *Costs* function were the only factor taken into account, the apartment Ap_F would have been chosen — note that $Cost(Ap\_B, Ap\_F) = 0.09768$ and $Cost(Ap\_F, Ap\_B) = 0.09760$. Nevertheless, by considering the other two heuristics we are adopting, we have a different result. The costs of Ap_B and Ap_F are almost equal, but Ap_B is the least extreme option, while Ap_F is the most extreme: Ap_F has the best values for some attributes (*uni*, *zone* and *stars*), but a high penalisation for others (*station* and *price*). Moreover, by analysing the cost-benefit relationship between options, we identify that the relationship between Ap_F and Ap_B, which is 0.999, is worse than the average 0.922. Table 10 shows the values of all functions calculated for every pair of options of the *Acceptable* set, and by considering the weighted sum of the costs, trade-off contrast and extremeness aversion, the **chosen option is** Ap_B.

Table 11: Complexity Analysis of our Technique.

Pre-processing	
PSM	$O(\|Opt\|\|P_e\|)$
OAPM	$O(\|Opt\|^2\|Att\|\|P_e\|)$
AVPO	$O(\|Opt\|^2 + \|P_e\|^2)$
Explication	$O(\|Opt\|^2\|Att\|\|P_e\|)$
Elimination	$O(\|Opt\|^2 + \|Opt\|\|Att\|)$
Selection	
Cost	$O(\|Opt\|^2\|Att\| + max(PP_i)\|PP_i\| + \|P_i\|\|Att\|)$
Extremeness Aversion	$O(\|Opt\|\|Att\| + \|Opt\|^2)$
Trade-off Contrast	$O(\|Opt\|^2)$
Decision Function	$O(\|Opt\|^2)$

Finally, we discuss the complexity of our technique. As it can be observed in the presented algorithms, our technique runs in polynomial time, and most of the algorithms require comparing each pair of options according to each attribute. We present the complexity of each part of our technique in Table 11, whose total is

$$O(|Opt|^2|Att||P_e| + |P_e|^2 + max(PP_i)|PP_i| + |P_i||Att|) \quad (11)$$

where Opt is the set of available options, Att is the set of attributes, P_e is the set of preferences, PP_i is the set of preference priorities, $max(PP_i)$ is the maximum number associated with the preference priorities, and P_i is the set of attribute priorities and attribute indifferences.

5. Evaluation

In this section, we evaluate our approach through an empirical evaluation, which compares our choices with those of a human expert, and with a user study. This allowed us to identify strengths and weaknesses of our approach. As the input of our technique is high-level preferences and existing approaches cannot handle all of them, our evaluation does not make side-by-side comparison with existing work, which is discussed in next section.

5.1. Empirical Evaluation

The empirical evaluation of our technique is based on the study (Nunes et al., 2010) that also informed the preference language itself. Participants provided preference specifications (in natural language) for use by an individual to buy a laptop on their behalf. Both these individuals and the domain expert were given

Table 12: Analysis of Domain Expert and our Technique choices.

	Our Technique (F)	**Expert (F)**	**Our Technique (5)**	**Expert (5)**
Average	63.19	61.25	61.94	61.44
Standard Deviation	13.36	11.93	8.00	8.32
Minimum	47.68	44.80	50.72	47.12
Maximum	100.00	100.00	100.00	96.39

a laptop catalogue (with 144 laptops) from which to choose and rank up to five options. The three relevant parts of the study used for our evaluation are the initial preference specification, the user choices and the domain expert recommendation. We use these to compare our decisions against those of the user and domain expert. Similarly to how the domain expert recommendation was assessed in the study, we calculate a similarity score $SS \in [0, 100]$, comparing the recommendation with the user choice, with 100 indicating a match to user choice. SS takes into account the position in the rank of chosen laptops using a weighted average.

Using a graphical user interface developed to input preferences according to our preference language, we were able to store the preferences provided by 113 participants. Of the 192 user specifications, 79 (41%) use subjectivity or purpose, and therefore cannot be expressed in our language. For example, "*I'd like a laptop to carry on my backpack*." Moreover, of these 79, 9 have no expert recommendation, because they did not specify preferences, such as "*I would never delegate this task [buying a laptop] to another person*." For the remaining specifications, we applied our technique (which takes an average of 3.6026 $seconds$ on an Intel Core i5 2.30GHz, 8GB of RAM, with standard deviation 1.4051, to be executed for each request, with 144 laptops, and 61 attributes), and obtained average, standard deviation, minimum and maximum of SS, shown in Table 12. The label "F" means the similarity score considering the first expert choice and the first choices of our technique compared to the first user choice, while the label "5" means the comparison between the top choices (up to five). If the domain expert recommended x laptops, we use the first x choices of our technique. Even though our technique does not rank acceptable options, as we calculate a numeric value to compare them, we use these numbers to obtain the top choices.

The results show that the values obtained for the (human) domain expert and our technique are not so different — considering only the first choices, our technique has $M_{SS} = 63.19$ and the domain expert has $M_{SS} = 61.25$; i.e. our technique has a better average SS than the expert. The same occurs for the up to five choices. The difference between the obtained similarity scores is *significant* when comparing only first choices, as determined by a Wilcoxon Signed Ranks test —

Measured Variable	Question responded on a 7-point Likert scale from "strongly disagree" to "strongly agree"
Choice Evaluation	
Choice quality	This application made really good choices.
Trust in choice	I feel that this application is trustworthy.
Decision confidence	I am confident that the choice made is really the best choice for me.
Approach Evaluation	
Perceived usefulness	This application is competent to help me effectively make choices I really like.
Intention to purchase	I would accept this choice if given the opportunity.
Intention to return	If I had to search for a product online in the future and an application like this was available, I would be very likely to use it.
Intention to save effort in next visit	If I had a chance to use this application again, I would likely make my choice more quickly.
Enjoyment	I found my visit to this application enjoyable.
Satisfaction	My overall satisfaction with the application is high.

Table 13: Measured Variables — adapted (Chen and Pu, 2010).

$W(112) = 3711, p = 0.0497$ (F), and thus we reject a null hypothesis that domain expert and our technique choices are equal. However, this is not the case of the up to five choices: $W(112) = 3774, p = 0.1131$ (5). As a consequence, considering our experiment, *we can conclude that our technique makes choices at least as good as those of the domain expert*.

5.2. User Study

As the focus of our technique is users, we also evaluated it with a user study, in which participants are requested to imagine a situation in which they are going to buy a new mobile phone. In addition, in this scenario, they are provided with an intelligent system that will make a choice on their behalf and asks them to specify their preferences and restrictions over the mobile phone they want. Participants are able to specify their preferences using our language through an interface that has many features, such as choosing explanation types with radio buttons, then selecting preferences parameters with combo boxes, setting preference formulae in a similar way to specifying rules in e-mail clients, and so on. Before providing their preferences, the participants receive a brief tutorial on how to interact with the interface and explanations about the language constructions.

Based on the provided participants' preferences, we choose an option using our decision making technique, and present to the participant: (i) the chosen mobile phone; (ii) the next four mobile phones of the acceptable set ranked according to the decision function of our technique (so as to form five chosen options, which was an adequate number in our previous study); and (iii) the remaining mobile phones initially hidden, but the participants can see it upon request to analyse all the 191 available mobile phones. With this presented choice, participants are

asked to evaluate it by answering questions associated with measurement variables that part of an user evaluation framework of recommender systems (Chen and Pu, 2010), whose questions were adapted to our study. The questions associated with the choice made by our technique are shown in the "Choice Evaluation" part of Table 13. Finally, participants have to answer final questions that evaluate the approach as a whole, which are shown in the "Approach Evaluation" part of Table 13. All the steps of this study were supported by a developed application, which includes an implementation of our decision making technique.

Our study included 35 participants, which were selected using convenience sampling, by making invitations for volunteers by email to the social network of the researchers involved in this study. However, as participants were observed while taking part of the study, only participants in the same locations (in two different Brazilian cities) of the lead researcher could be selected.

By observing the interaction of participants with the interface, we noticed they first took a few moments to get familiar with it, and then explored the available preference types, priorities and attributes. Many participants began providing their preferences by specifying the attributes they do not care about, for later concentrating on the characteristics they desire. On average participants took 15*min* to specify their preferences, including the time to give the brief tutorial. Based on the provided preferences, participants had a choice made on their behalf, and they had to evaluate this choice (and other selected options that are close to the chosen option) with respect to the *choice quality*, *trust in choice* and *decision confidence* — also presented in Figure 3. It can be seen that our technique was evaluated with high levels of choice quality and trust, and also of decision confidence (but with a level not as high as the other measurements) indicating that our technique is able to make adequate choices. We identified three situations in which the decision made was not good. First, some participants specified preference order over some values, and did not specify that those values were actually preferred to other unmentioned values. Second, the same occurred when they specified preference indifferences. Third, there was one case where the participant provided a hard constraint that caused most of the models (which satisfy her other preferences) to be discarded, and received choices that did not satisfy her other preferences. An interesting situation is that some participants did not (strongly) agree with the decision, but they said they understood it. These participants realised that they forgot to specify something in their preferences as the first choices had undesired characteristics, which they did not mention as a preference.

In the last step of the experiment, participants had to answer questions whose goal is to evaluate the whole approach. Figure 3 depicts the results obtained,

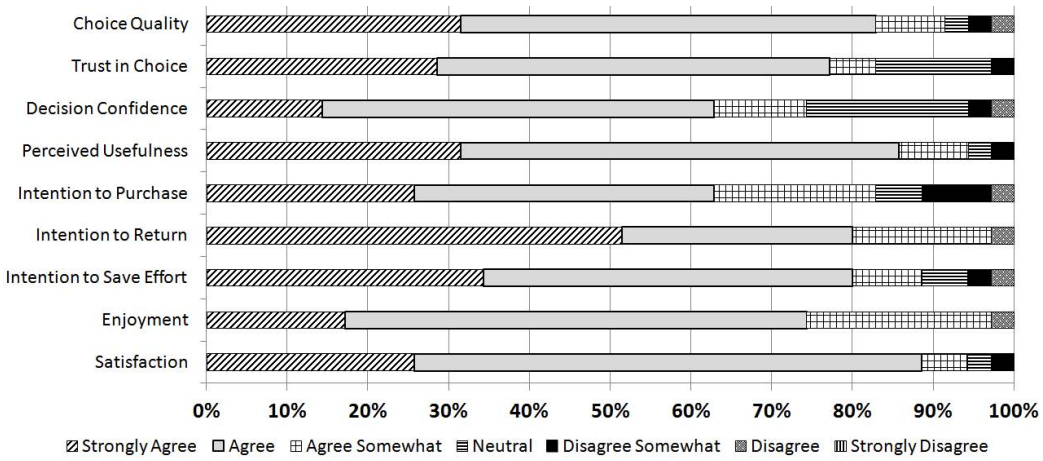

Figure 3: User study results.

showing that a representative amount of participants answered (strongly or somewhat) agree for the *perceived usefulness* (94.29%), *intention to return* (97.14%), *enjoyment* (97.14%) and *satisfaction* (94.28%). There were participants who declared that they indeed needed a mobile phone and they were happy with the system and the recommended choices. Two of the measurements do not follow this case: *intention to purchase* and *intention to save effort*. Although many participants stated that the choice quality was good (first five options), they were not sure or disagreed that the chosen option was the best for them. Not having a real intention to purchase also affected this measurement. Therefore, 17.14% of the participants answered neutral or disagree with respect to the intention to purchase. Regarding the intention to save effort, some participants claimed that even though using the system may help them with their choice, it would require more effort — but they are willing to use it anyway, because they believe it can help them to make a choice better than that they would do without support.

6. Related Work

Most existing work related to decision making is founded on Multi-Attribute Utility Theory (MAUT) (Keeney and Raiffa, 1976), which emphasises the use of multi-attribute preference models based on the theories of Neumann and Morgenstern (1944), who present a set of axioms for preferences and utilities such that any decision-maker satisfying these axioms has a *utility function (UF)*. As with UFs it is possible to order available options, thus choosing among them, different ap-

proaches (McGeachie and Doyle, 2004; Domshlak and Joachims, 2007) have been proposed to transform specific models that capture qualitative preferences (which are closer to how users express preferences) into UFs, i.e. quantitative preferences, which are consistent with the constraints established by the qualitative preferences. Some approaches (Bistarelli et al., 1997, 2010; Gelain et al., 2010) extend Constraint Satisfaction Problems (CSPs) to incorporate *soft constraints* (that can remain unsatisfied), namely Soft Constraint Satisfaction Problem (SCSP), associating a penalty (or preference) with each constraint, and creating an optimisation problem of minimising penalty (or maximising preference). Utility functions and SCSPs are classical approaches for dealing with preferences and making decisions, but the former are hard to elicit and the latter deal with over-constrained problems rather than choosing from feasible solutions.

A third group of approaches, mainly represented by CP-Nets (Boutilier et al., 2004) and TCP-Nets (Brafman et al., 2006), takes another direction, proposing new *graphical structures* to represent and reason about qualitative preferences. Finally, work in the area of databases proposes *extensions of query languages* (Agrawal and Wimmers, 2000; Chomicki, 2003; Kieling, 2002; Koutrika and Ioannidis, 2006; Siberski et al., 2006) to incorporate preferences and algorithms to provide query results according to specified preferences. Even though these approaches propose different solutions, they share the common goal of making a choice based on preferences. However, as shown in Table 14, they address limited kinds of preferences, restricting their natural expression by humans. Our technique is the only one that at exploits natural language expressions — expressive speech acts — to make decisions on behalf of users.

Besides being able to deal with restricted kinds of preferences in comparison to our technique, these approaches only choose between two options when the preferences provided are sufficient to make the decision, i.e. if the decision involves trade-off, users must have previously resolved it and specified their preferences. However, as mentioned in the introduction, as discussed by Tversky (1996), people resolve trade-offs in light of available options, and do not provide such preferences. Our technique, on the other hand, resolves trade-offs using (i) preferences over individual attributes; (ii) priorities; and (iii) psychology-inspired heuristics, with the aim of performing a reasoning similar to humans.

7. Final Considerations

In this paper, we proposed an automated decision making technique that uses preferences expressed by users in a high-level language, which is close to how

Table 14: Reasoning Approaches vs. Preferences.

Approach	Preference								Priority		
	Cd	Ct	G	O	Q	R	I	D	a	i	p
UF-based (McGeachie and Doyle, 2004)				X							
SVM-based (Domshlak and Joachims, 2007)				X		X	X				
SCSP (Bistarelli et al., 1997)		X									X
Bipolar preferences (Bistarelli et al., 2010)		X									X
Interval-valued SCSP (Gelain et al., 2010)		X									X
CP-Nets (Boutilier et al., 2004)	X			X							
TCP-Nets (Brafman et al., 2006)	X			X					X		
Scoring Function (Agrawal and Wimmers, 2000)		X				X		X			X
Winnow (Chomicki, 2003)		X		X			X				X
Best-Matches-Only (BMO) (Kieling, 2002)		X		X		X					X
Query Personalisation (Koutrika and Ioannidis, 2006)						X					X
SPARQL (Siberski et al., 2006)		X	X								X

Legend — Cd: condition; Ct: constraint; G: goal; O: order; Q: qualifying; R: rating; I: indifference; D: don't care; a: attribute priority; i: attribute indifference; p: preference priority.

people express their known preferences in natural language. The technique resolves trade-offs based on priorities, which indicate attributes they consider more important, combined with psychology-inspired heuristics, thus making decisions in way similar to how humans do, with the aim of automating tasks on their behalf. Our technique provides the novelty of exploiting different natural language expressions and psychology-inspired heuristics in automated decision making. These two particularities of our technique consist of two ways of significantly improving research in this area: while expressive speech acts and other expressions give valuable information that can be used to generate low-level preference representations, such as utility functions; psychology-inspired heuristics can be used to reduce the amount of preferences obtained from users, as they can predict how users would resolve trade-offs.

The empirical evaluation of our technique showed that it is able to make a choice on behalf of the users at least as good as that made by a human domain expert, considering our experiment. However, the user study performed indicated that, even though our technique indicates good recommendations and helps users to make choices, it is not always able to make the right choice on their behalf. Nevertheless, because our technique is inspired by how humans make decisions, we may produce a reasoning process that can be analysed to generate explanations why a certain option was chosen, and why the remaining ones were rejected. This may help users to further refine their preferences to obtain a better choice and also increase the levels of trust in choice and decision confidence of our decision making technique. As our decision making technique has variable parts (e.g. functions and weights), which were instantiated in this paper after running the technique

with different alternatives, it is our future work is to improve results by exploring this variability. Examples are using natural language processing and fuzzy logic to interpret modifiers, and machine learning to calibrate selected functions.

Appendix A. Algorithms

Algorithm 1: PSM Builder

Input: MP: monadic preferences; Opt: options; Att: attributes; $scale$: modifier scale
Output: PSM: preference satisfaction model

```
1  foreach Option o ∈ O do
2      foreach Attribute a ∈ A do
3          x ← null;
4          m* ← null;
5          foreach ⟨ϵ, m⟩ ∈ AttMod(MP, o, a) do
6              if m* == null ∨ |IndexOf(m, scale)| > |IndexOf(m*, scale)| then
7                  x = ϵ;
8                  m* ← m;
9          if m* == null then
10             foreach ⟨¬, m⟩ ∈ AttMod(MP, o, a) ∧ Positive(m) do
11                 if m* == null ∨ IndexOf(m, scale) < IndexOf(m*, scale) then
12                     x = ¬;
13                     m* ← m;
14         if m* == null then
15             foreach ⟨¬, m⟩ ∈ AttMod(MP, o, a) do
16                 if m* == null ∨ IndexOf(m, scale) > IndexOf(m*, scale) then
17                     x = ¬;
18                     m* ← m;
19         PSM[o, a] ← ⟨x, m*⟩;
20 return PSM;
```


Adomavicius, G., Tuzhilin, A., june 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on 17 (6), 734–749.

Agrawal, R., Wimmers, E. L., 2000. A framework for expressing and combining preferences. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. ACM, New York, NY, USA, pp. 297–306.

Bistarelli, S., Montanari, U., Rossi, F., March 1997. Semiring-based constraint satisfaction and optimization. J. ACM 44, 201–236.

Algorithm 2: AVPO Builder

Input: o: option; a: attribute; *Order*: order preferences
Output: $AVPO$: $\langle N, A \rangle$

```
1 Set⟨AVPONode⟩ N ← ∅;
2 Set⟨Arrow⟩ A ← ∅;
3 foreach Order preference op ∈ Order do
4     if App(op, o) ∧ att(op) = a then
5         N ← N ∪ { LHS(op), RHS(op) };
6         A ← A ∪ {⟨ LHS(op), RHS(op) ⟩};
7 return ⟨N, A⟩;
```

Algorithm 3: $LastEqual(tag, node, up, scale)$

Input: *tag*: modifier to be searched for;*node* : *AVPONode*; *up* : *boolean* (flag to indicate if the search should be in the parents or the children of *node*; *scale*: modifier scale
Output: $\langle firstTagged, dist \rangle : \langle AVPONode, int \rangle$, first tagged node and the distance from it to *node*

```
1  List⟨ AVPONode ⟩ nodeList ← up ? Parents(node) : Children(node);
2  ⟨ AVPONode, int ⟩ lastEqual ← null;
3  double tagValue ← f_m(IndexOf(tag, scale));
4  foreach AVPONode next ∈ nodeList do
5      double nextTagValue ← null;
6      if Tag(next) ≠ null then
7          double nextTagValue ← f_m(IndexOf(Tag(next), scale));
8      if nextTagValue = null ∨ tagValue = nextTagValue then
9          ⟨ AVPONode, int ⟩ temp ← LastEqual(tag, next, up, scale);
10         if temp ≠ null then
11             temp_2 ← π_2(temp) + 1;
12         else if tagValue = nextTagValue then
13             temp ← ⟨next, 0⟩;
14         if lastEqual = null ∨ π_2(lastEqual) < π_2(temp) then
15             lastEqual ← temp;
16 return firstTagged;
```

Algorithm 4: $TaggedNodeValue(node, scale)$

Input: $node$: $AVPONode$; $scale$: modifier scale
Output: $value$: $double$

```
1  int dist ← 0;
2  ⟨ AVPONode, int ⟩ above ← LastEqual(Tag(node), node, true, scale);
3  if above ≠ null then
4      dist ← π2(above);
5  ⟨ AVPONode, int ⟩ below ← LastEqual(Tag(node), node, false, scale);
6  if below ≠ null then
7      dist ← π2(below);
8  double tagValue ← fm(IndexOf(Tag(node),scale));
9  if dist = 0 then
10     return tagValue;
11 else
12     double max ← tagValue;
13     double temp ← fm(IndexOf(Tag(node), scale) +1);
14     if temp ≠ null then
15         max = max + |temp − tagValue|/2;
16     double min ← tagValue;
17     double temp ← fm(IndexOf(Tag(node), scale) −1);
18     if temp ≠ null then
19         min = min − |temp − tagValue|/2;
20 if above = null then
21     return max;
22 else
23     double step ← |max − min|/dist;
24     return max − π2(above) × step;
```

Algorithm 5: $FirstTaggedNode(node, up, scale)$

Input: $node$: $AVPONode$; up : $boolean$ (flag to indicate if the search should be in the parents or the children of $node$; $scale$: modifier scale
Output: $\langle firstTagged, dist\rangle$: $\langle AVPONode, int\rangle$, first tagged node and the distance from it to $node$

```
1  List⟨ AVPONode ⟩ nodeList ← up ? Parents(node) : Children(node);
2  ⟨ AVPONode, int ⟩ firstTagged ← null;
3  double rate ← 0;
4  foreach AVPONode next ∈ nodeList do
5      ⟨ AVPONode, int ⟩ temp ← null;
6      if Tag(next) = null then
7          temp ← FirstTaggedNode(next, up, scale);
8      else
9          temp ← ⟨next, 0⟩;
10     temp_2 ← π_2(temp) + 1;
11     double tagValue ← f_m(IndexOf(Tag(π_1(temp)), scale));
12     double step ← |tagValue|/π_2(temp);
13     if up then
14         double rateTemp ← tagValue − (π_2(temp) − 1) × step;
15         if rate = null ∨ rateTemp < rate then
16             firstTagged ← temp;
17             rate ← rateTemp;
18     else
19         double rateTemp ← tagValue + (π_2(temp) − 1) × step;
20         if rate = null ∨ rateTemp > rate then
21             firstTagged ← temp;
22             rate ← rateTemp;
23 return firstTagged;
```

Algorithm 6: $UntaggedNodeValue(node, up, scale)$

Input: $node$: $AVPONode$; $scale$: modifier scale
Output: $value$: $double$

```
1  ⟨ AVPONode, int ⟩ above ← FirstTaggedNode(node, true, scale);
2  double aboveValue ← TaggedNodeValue(π_1(above), scale);
3  ⟨ AVPONode, int ⟩ below ← FirstTaggedNode(node, false, scale);
4  double belowValue ← TaggedNodeValue(π_1(below), scale);
5  double step ← |aboveValue − belowValue|/(π_2(above) + π_2(below));
6  return aboveValue − (π_2(above) × step);
```

Algorithm 7: *ProcessPreferencePriorities*(*priorities*, *allAtt*)

Input: *priorities*: preference priorities applicable to an option; *allAtt*: set of attributes
Output: *attPO*: attribute partial order

```
1  Set⟨Attribute⟩ N ← ∅;
2  Set⟨Arrow⟩ A ← ∅;
3  Set⟨Attribute⟩ parents ← ∅;
4  int i ← 1;
5  while priorities ≠ ∅ do
6      Set⟨Attribute⟩ currentAtt ← ∅;
7      while ∃pri.(pri ∈ priorities∧ Priority(pri) = i) do
8          PreferencePriority pri ← Get(priorities, i);
9          Attribute a ← Attribute(pri);
10         if a ∉ N then
11             N ← N ∪ {a};
12             currentAtt ← currentAtt ∪ {a};
13         priorities = priorities\{pri};
14     if currentAtt ≠ ∅ then
15         foreach a ∈ currentAtt do
16             foreach p ∈ parents do
17                 A ← A ∪ {⟨p, a⟩};
18         parent ← currentAtt;
19     i ← i + 1;
20 foreach a ∈ (allAtt\N) do
21     N ← N ∪ {a};
22     foreach p ∈ parents do
23         A ← A ∪ {⟨p, a⟩};
24 return ⟨N, A⟩;
```

Algorithm 8: *MoveAbove*(att_1, att_2, A)

Input: att_1, att_2: attributes to be swapped; A: attPO arrows

```
1  Set⟨Attribute⟩ oldParents ← Parents(att1) ;
2  Set⟨Attribute⟩ oldChildren ← Children(att1) ;
3  foreach p ∈ oldParents do
4      foreach c ∈ oldChildren do
5          A ← A ∪ {⟨p, c⟩};
6      A ← A\{⟨p, att1⟩};
7  foreach c ∈ oldChildren do
8      A ← A\{⟨att1, c⟩};
9  foreach p ∈ Parents(att2) do
10     A ← A ∪ {⟨p, att1⟩};
11     A ← A\{⟨p, att2⟩};
12 A ← A ∪ {⟨att1, att2⟩};
13 foreach p ∈ oldParents do
14     if ¬ExistsPath(att1, p) ∧ ¬ExistsPath(p, att1) then
15         A ← A ∪ {⟨p, att1⟩};
16 foreach c ∈ oldChildren do
17     if ¬ExistsPath(att1, c) ∧ ¬ExistsPath(c, att1) then
18         A ← A ∪ {⟨att1, c⟩};
```


Bistarelli, S., Pini, M. S., Rossi, F., Venable, K. B., June 2010. From soft constraints to bipolar preferences: modelling framework and solving issues. J. Exp. Theor. Artif. Intell. 22, 135–158.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., Poole, D., 2004. Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Int. Res. 21 (1), 135–191.

Brafman, R. I., Domshlak, C., Shimony, S. E., March 2006. On graphical modeling of preference and importance. J. Artif. Int. Res. 25, 389–424.

Chen, L., Pu, P., 2010. User evaluation framework of recommender systems. In: Proceedings of 2010 Workshop on Social Recommender Systems (SRS'10) at the ACM International Conference on Intelligent User Interfaces (IUI'10). ACM, New York, NY, USA.

Chomicki, J., December 2003. Preference formulas in relational queries. ACM Trans. Database Syst. 28, 427–466.

Algorithm 9: *MoveEqual*(att_1, att_2, A)

Input: att_1, att_2: attributes to be swapped; A: attPO arrows

```
1  Set⟨Attribute⟩ oldParents ← Parents(att1) ;
2  Set⟨Attribute⟩ oldChildren ← Children(att1) ;
3  foreach p ∈ oldParents do
4      foreach c ∈ oldChildren do
5          A ← A ∪ {⟨p, c⟩};
6      A ← A\{⟨p, att1⟩};
7  foreach c ∈ oldChildren do
8      A ← A\{⟨att1, c⟩};
9  foreach p ∈ Parents(att2) do
10     A ← A ∪ {⟨p, att1⟩};
11 foreach c ∈ Children(att2) do
12     A ← A ∪ {⟨att1, c⟩};
13 foreach p ∈ oldParents do
14     if ¬ExistsPath(att1, p) ∧ ¬ExistsPath(p, att1) then
15         A ← A ∪ {⟨p, att1⟩};
16         A ← A ∪ {⟨p, att2⟩};
17 foreach c ∈ oldChildren do
18     if ¬ExistsPath(att1, c) ∧ ¬ExistsPath(c, att1) then
19         A ← A ∪ {⟨att1, c⟩};
20         A ← A ∪ {⟨att2, c⟩};
```

Cormen, T. H., Stein, C., Rivest, R. L., Leiserson, C. E., 2001. Introduction to Algorithms, 2nd Edition. McGraw-Hill Higher Education.

Domshlak, C., 2008. A snapshot on reasoning with qualitative preference statements in ai. In: Maier, G., Rammerstorfer, F. G., Salenon, J., Schrefler, B., Serafini, P., Riccia, G., Dubois, D., Kruse, R., Lenz, H.-J. (Eds.), Preferences and Similarities. Vol. 504 of CISM International Centre for Mechanical Sciences. Springer Vienna, pp. 265–282.

Domshlak, C., Joachims, T., 2007. Efficient and non-parametric reasoning over user preferences. User Modeling and User-Adapted Interaction 17 (1-2), 41–69.

Gelain, M., Pini, M. S., Rossi, F., Venable, K. B., Wilson, N., April 2010. Interval-valued soft constraint problems. Annals of Mathematics and Artificial Intelligence 58, 261–298.

Greco, S., 2005. Multiple Criteria Decision Analysis: State of the Art Surveys. Springer Science + Business Media, Inc.

Hansson, S. O., 1990. Defining "good" and "bad" in terms of "better". Notre Dame Journal of Formal Logic 31 (1), 136–149.

Hansson, S. O., 2001. The Structure of Values and Norms (Cambridge Studies in Probability, Induction and Decision Theory). Cambridge University Press.

Kaklauskas, A., Zavadskas, E. K., Trinkunas, V., 2007. A multiple criteria decision support on-line system for construction. Engineering Applications of Artificial Intelligence 20 (2), 163–175.

Keeney, R. L., Raiffa, H., 1976. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley series in probability and mathematical statistics. John Wiley & Sons, Inc, New York.

Kieling, W., 2002. Foundations of preferences in database systems. In: Proceedings of the 28th international conference on Very Large Data Bases. VLDB '02. VLDB Endowment, pp. 311–322.

Klein, D. A., Shortliffe, E. H., June 1994. A framework for explaining decision-theoretic advice. Artif. Intell. 67, 201–243.

Koutrika, G., Ioannidis, Y., 2006. Personalization of queries based on user preferences. In: Bosi, G., Brafman, R. I., Chomicki, J., Kießling, W. (Eds.), Preferences: Specification, Inference, Applications. No. 04271 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany.

Labreuche, C., May 2011. A general framework for explaining the results of a multi-attribute preference model. Artif. Intell. 175, 1410–1448.

Lichtenstein, S., Slovic, P., 2006. The construction of preference. Cambridge University Press, New York, USA.

McGeachie, M., Doyle, J., 2004. Utility functions for ceteris paribus preferences. Computational Intelligence 20 (2), 158–217.

Neumann, J. V., Morgenstern, O., 1944. Theory of Games and Economic Behavior. Princeton University Press.

Nunes, I., Barbosa, S., Lucena, C., October 2010. Understanding how users express preferences: a user study. Tech. Rep. CS-2010-19, University of Waterloo, Waterloo, Canada.

Nunes, I., Barbosa, S. D., Cowan, D., Miles, S., Luck, M., de Lucena, C. J., 2013. Natural language-based representation of user preferences. Interacting with Computers.

Nunes, I., Miles, S., Luck, M., Barbosa, S. D. J., de Lucena, C. J. P., 2014. Pattern-based explanation for automated decisions. In: ECAI 2014 - 21st European Conference on Artificial Intelligence. pp. 669–674.

Schwartz, B., 2005. The paradox of choice: Why more is less. Harper Perennial.

Shafir, E., Simonson, I., Tversky, A., 1998. Reason-based choice. In: Preference, Belief and Similarity. MIT Press, pp. 937–962.

Siberski, W., Pan, J. Z., Thaden, U., 2006. Querying the semantic web with preferences. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L. (Eds.), The Semantic Web - ISWC 2006. Vol. 4273 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 612–624.

Simonson, I., Tversky, A., 1992. Choice in context: Tradeoff contrast and extremeness aversion. Journal of Marketing Research 29 (3), 281–295.

Tversky, A., 1996. Contrasting rational and psychological principles of choice. In: Zeckhauser, R. J., Keeney, R. L., Sebenius, J. K. (Eds.), Wise Choices : Games, Decisions, and Negotiations. Harvard Business School Press, pp. 5–21.

Wan, S., Dong, J., 2014. A possibility degree method for interval-valued intuitionistic fuzzy multi-attribute group decision making. Journal of Computer and System Sciences 80 (1), 237–25.

Wan, S.-P., Li, D.-F., 2013. Fuzzy {LINMAP} approach to heterogeneous {MADM} considering comparisons of alternatives with hesitation degrees. Omega - The International Journal of Management Science 41 (6), 925–940.

Wierenga, B., 2008. Handbook of Marketing Decision Models, 1st Edition. Springer Publishing Company, Incorporated.